– Technischer Bericht 372/09 –

Duty Cycling in drahtlosen Multi-Hop-Netzwerken

Dennis Christmann (christma@cs.uni-kl.de)

Zusammenfassung: Durch aktuelle Entwicklungen, Geräte immer häufiger ausschließlich mit Batterien zu betreiben, erfährt Energieeinsparung im Allgemeinen und Duty Cycling im Speziellen eine immer größere Bedeutung. Dieser Bericht stellt eine Diskussion vorhandener Protokolle der MAC-Ebene dar, die sich dem Problem des Duty Cyclings annehmen. Der Fokus liegt dabei auf unkoordinierten Sensor-Netzwerken und den Möglichkeiten zum Einsparen von Energie durch Deaktivieren des Transceivers. Neben der Vorstellung von bekannten Ansätzen, wie S-MAC und T-MAC, werden auch die Vor- und Nachteile der Lösungen analysiert und qualitativ verglichen. Dabei steht vor Allem der Kompromiss zwischen Energieeinsparungen und klassischen Dienstgütecharakteristika wie Ende-zu-Ende-Verzögerungen im Mittelpunkt. Des Weiteren gibt der Bericht einen Überblick über die Vielfalt weiterer Lösungsideen.

Inhaltsverzeichnis

1 Einführung

Einhergehend mit der zunehmenden Bedeutung und Präsenz von eingebetteten Systemen ist die Verringerung des Energieverbrauchs ein Thema, welches immer mehr an Relevanz gewinnt. Häufig werden derartige Systeme in Szenarien eingesetzt, bei denen eine permanente Energieversorgung der einzelnen Knoten nicht möglich ist und die Knoten stattdessen mit Batterien ausgestattet werden müssen. Diese Einschränkungen findet man vor allem in drahtlosen Sensornetzwerken (*Wireless Sensor Networks*; WSN), die aus sehr vielen kleinen Knoten bestehen, deren Ressourcen oft stark limitiert sind [2]. Sensornetzwerke haben bereits heute Einzug in zahlreiche medizinische und militärische Anwendungsgebieten gefunden oder werden in Bereichen der Hausautomation oder in Katastrophenszenarien, wie z.B. bei Waldbränden, eingesetzt. Oft können in diesen Szenarien die Energiequellen nur schwer oder gar nicht ausgetauscht werden, sodass die Lebensdauer eines einzelnen Knotens von dessen Energieverbrauch abhängt. Aufgrund der Eigenschaft von WSNs, dass Informationen häufig über mehrere Hops transportiert werden und an eine Senke adressiert sind, bestimmt die Lebensdauer einzelner Knoten nicht selten die Lebensdauer des ganzen Systems [24].

Diese Ausarbeitung beschäftigt sich mit dem Problem des *Duty Cyclings* in nicht-koordinierten drahtlosen Netzwerken. Duty Cycling basiert auf der Annahme, dass einzelne Knoten nicht ununterbrochen an einer Kommunikation beteiligt sind. Diese Annahme gilt besonders in WSNs, bei denen für die Anwendung interessante Ereignisse nur selten auftreten und das Nachrichtenaufkommen gering ist [11]. Dadurch können Knoten zeitweise Teile der Hardware abschalten bzw. in einen eingeschränkten Modus („Schlafmodus“) wechseln, um den Energieverbrauch zu reduzieren (siehe Abbildung 1). Die Herausforderung dabei ist, die Kommunikation zwischen zwei Knoten trotz Schlaf-Phasen zu ermöglichen. Dazu müssen zwei Knoten, die miteinander kommunizieren wollen, zum gleichen Zeitpunkt in der Aktiv-Phase sein, damit ein Knoten die Übertragung eines sendenden Knotens hören kann. Die Aktiv- und Schlaf-Phase wird in mancher Literatur als *Frame* bezeichnet [32]. Als Duty Cycle bezeichnet man das Verhältnis von Aktiv-Phase zur gesamten Framedauer.

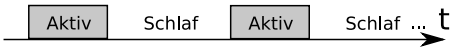

Abbildung 1: Konzept von Duty-Cycling [31].

Der Fokus der Ausarbeitung liegt auf Duty Cycling auf MAC-Ebene (Medium Access Control), d.h. wie bereits bei dem Zugriff auf das Medium Energie eingespart werden kann, indem Zeitintervalle für den Schlafmodus definiert werden. Die Ausarbeitung beschränkt sich zudem auf Energieeinsparungen durch Wechseln des Transceivers zwischen verschiedenen Modi. Es werden keine weiteren Einsparungsmöglichkeiten durch Ausschalten von angeschlossenen Sensoren oder der Hardware-Plattform betrachtet. Neben den behandelten Lösungen auf MAC-Ebene, kann Energieeffizienz allgemein auch auf anderen Protokollschichten erhöht werden. Z.B. kann die Netzwerkebene die Lebensdauer des Systems erhöhen, indem Fluten des Mediums vermieden und Energieeffizienz zu einem Kriterium bei dem Finden von Routen wird [2, 27]. Aber auch die Anwendungsebene kann zu dem Einsparen von Energie beitragen, indem sie Informationen über die zu erwartende Kommunikation liefert und dadurch Duty Cycling bei der Erstellung von optimalen Schedules unterstützt.

Bei dem Entwurf von MAC-Protokollen für WSNs ist – neben den bekannten Dienstgütekriterien wie Ende-zu-Ende-Verzögerung, Übertragungsrate und dem Quotienten aus gesendeten und empfangenen Daten (Packet Delivery Ratio - PDR) – die Verringerung des Energieverbrauchs ein immer bedeutenderer Faktor [27]. Allerdings sind Dienstgütekriterien im üblichen Sinne und Verringerung des Energieverbrauchs gegensätzliche Ziele. Durch Duty Cycling wird der Verbrauch verringert, indem der Transceiver eines Knotens möglichst lange in einen energiesparenden Modus versetzt wird. Dies führt allerdings unmittelbar zu einer Erhöhung der Ende-zu-Ende-Verzögerung, da jeder sendende Knoten mit dem Sendebeginn warten muss, bis der Kommunikationspartner die Schlaf-Phase verlässt und wieder empfangsbereit ist. Insbesondere in zeitkritischen Anwendungen, wie z.B. einem Feuermeldesystem, darf diese Verzögerung nicht zu groß werden [15]. Zusätzlich steigt bei wettbewerbsbasiertem Mediumzugriff die Kollisionswahrscheinlichkeit je kleiner die aktiven Zeitabschnitte gewählt werden [7]. Das Ziel ist es demnach, einen Kompromiss zu finden und Schlaf-Phasen so lange wie möglich zu wählen, ohne dabei die Dienstgüte inakzeptabel zu verschlechtern [27, 20].

Im Unterschied zu den Problemen in unkoordinierten Drahtlosnetzen, lässt sich Energieeffizienz in koordinierten Drahtlosnetzen wie IEEE 802.15.4 [13] oder Bluetooth [10] besser umsetzen, da hier ein Koordinator individuell die Schlafzyklen einzelner Stationen bestimmen kann [20]. Dadurch wissen Knoten vorab, wann Kommunikation stattfindet, in welcher sie nicht beteiligt sind, und können in den Schlafmodus wechseln. Zusätzlich ist die Kollisionswahrscheinlichkeit bei koordinierten Protokollen geringer, da der Mediumzugriff

durch den Koordinator reguliert wird. Allerdings haben existierende koordinierte MAC-Protokolle (insbesondere 802.15.4 [13]) den Nachteil, dass bei der Multi-Hop-Übertragung von geringen Datenmengen die Ende-zu-Ende-Verzögerung höher als bei unkoordinierten MAC-Protokollen ist [20, 16]. Auch Bluetooth [10] ist durch notwendiges Clustern und die Anforderungen an eine sehr genaue Synchronisation für WSNs weniger geeignet [30]. Zusätzlich bieten auf TDMA basierende Protokolle wie Bluetooth schlechte Skalierungseigenschaften, da beim Eintreten neuer Knoten die Zeit-Slots neu verteilt werden müssen [31, 32], bzw. – wie im Falle von Bluetooth – die Anzahl der Stationen pro Cluster stark beschränkt ist. Das Problem der bestehenden koordinierten Protokolle lässt sich zusammenfassen auf die fehlende Fähigkeit, Multi-Hop-Netzwerke mit einem einzigen Koordinator zu verwalten. Dadurch wird eine Inter-Cluster-Koordination notwendig, die verhindern muss, dass sich zwei benachbarte Netzwerke bei der Intra-Cluster-Kommunikation behindern und trotzdem einen Austausch von Informationen zwischen Clustern ermöglicht [31, 32].

Die folgende Ausarbeitung beschäftigt sich zunächst in Abschnitt 2 mit den Ursachen von vermeidbarem Energieverbrauch auf MAC-Ebene. Abschnitt 3 stellt anschließend einige MAC-Protokolle vor, die eine Verringerung des Energieverbrauchs erreichen. Zum Schluss fasst der letzte Abschnitt 4 die Inhalte zusammen.

2 Energieverbrauch in Drahtlos-Netzwerken

2.1 Energieverbrauch auf Sensorknoten

Bevor eine Betrachtung der Ursachen von vermeidbarem Energieverbrauch folgt, werden zunächst die Energieverbraucher eines Netzwerknotens identifiziert. (Sensor)-Knoten bestehen aus mehreren Hardware-Komponenten, die jeweils getrennt voneinander Leistung aufnehmen und zum Gesamtverbrauch des Knotens beitragen [3]. Hierzu zählen unter Anderem der Transceiver und Sensoren.

Modus	**Stromstärke**
Mikrocontroller (schlafend)	$<15\,\mu$A
Mikrocontroller (aktiv)	8 mA
Transceiver (schlafend)	$426\,\mu$A
Transceiver (sendend)	17,4 mA
Transceiver (empfangend)	18,8 mA
Licht-/Spannungswandler	1.1 mA

Tabelle 1: Energieverbrauch [4, 28, 26].

Im Folgenden wird die Größenordnung der aufgenommenen Leistung am Beispiel eines MicaZ-Knotens betrachtet [6], um das Potential von Einsparungen zu verdeutlichen. Der betrachtete MicaZ-Knoten besitzt einen CC2420 Transceiver [28], einen ATMEL Atmega128L Mikrocontroller [4] und einen Lichtsensor [26]. Tabelle 1 zeigt in Ausschnitten die verbrauchte Strommenge der Komponenten mit den Daten aus den jeweiligen Datenblättern. Mit einer Spannung von ca. 3 V lässt sich die verbrauchte Leistung berechnen, die proportional zur Stromstärke ist.

Die Tabelle zeigt, dass der Transceiver den größten Anteil des Energieverbrauchs ausmacht. Außerdem ist erkennbar, dass die Energieaufnahme durch alleiniges Verringern der Anzahl an Übertragungen nicht reduziert werden kann, da der Transceiver im Empfangsmodus mehr Leistung aufnimmt als im Sendemodus. Abhängig von der Art der verwendeten Sensoren können auch Sensoren einen großen Teil des Energieverbrauchs ausmachen [3].

Der Rest der Ausarbeitung befasst sich allerdings nur mit möglichen Einsparungen durch Abschalten des Transceivers, da weitere Einsparungen meist von der Art der Anwendung abhängen. Dazu werden zunächst Quellen von 'Energieverschwendung' identifiziert, d.h. Szenarien, in denen einzelne Knoten aktiv sind (bzw. sein müssen) ohne dass sie relevante Informationen empfangen oder senden.

2.2 Quellen von Energieeffizienz-Minderung

2.2.1 Idle Listening

Unter *Idle Listening* versteht man das Abhören des Mediums im Empfangsmodus, ohne dass Daten empfangen werden [31, 32, 19, 27, 29]. Sie ist die größte Quelle von Energie-Ineffizienz; insbesondere da in Sensornetzwerken in der Regel nur sehr wenige Informationen versendet werden. Das Problem ist allgemeiner Natur in verteilten Systemen, da das Fehlen von globalem Wissen in der Regel verhindert, dass eine Station genau dann aufwachen kann, wenn einer anderen Station Daten zur Übertragung vorliegen. Ausnahmen bilden nur streng periodische Verkehrsmuster mit reservierungsbasierten Übertragungen. Allerdings sind diese Voraussetzungen in Ad-Hoc-Netzwerken, wie sie in der Ausarbeitung behandelt werden, in der Regel nicht gegeben.

2.2.2 Kollisionen

Zwei gleichzeitig gesendete Rahmen können in drahtlosen Netzwerken zu einer Kollision führen, die ein korrektes Empfangen verhindert [31, 32, 29]. Dies verschlechtert in zweierlei Hinsicht die Energieeffizienz: Zunächst waren Stationen aktiv, ohne dass sie effektiv kommuniziert haben. Zusätzlich muss die Übertragung oftmals wiederholt werden, sodass die Stationen weitere Zeit aktiv sein müssen.

2.2.3 Overhearing

Overhearing beschreibt das Mithören von Übertragungen, die an einen anderen Empfänger gerichtet sind [31, 32, 27, 29]. Insbesondere in drahtlosen Netzwerken besteht dieses Problem, da Kommunikation über ein Broadcastmedium stattfindet. Eine Station, die eine fremde Übertragung hört, kann während der Übertragung weder weitere Rahmen empfangen, noch eigene Daten senden, da nicht sichergestellt werden kann, dass die Übertragungen auf den Empfangsseiten kollidieren.

2.2.4 Overhead durch Kontrollnachrichten

Overhead durch Kontrollnachrichten kann bedingt als Quelle von Energie-Ineffizienz angesehen werden [31, 32, 29]. Kontrollnachrichten beinhalten zwar meist keine Informationen, an denen die Applikationen auf den (Sensor)-Knoten unmittelbar Interesse haben, sind aber notwendig, um Energieverschwendung durch die bereits genannten Quellen zu verringern. Als Beispiel seien hier RTS/CTS-Rahmen genannt [5], deren Intention die Verminderung von Kollisionen ist, die durch das „Hidden Station"-Problem bedingt sind.
Weiterer Overhead von Kontrollinformationen entsteht durch Header und Trailer, die von mehreren Layern eines Protokoll-Stacks zu den Informationen der Anwendungsschicht ergänzt werden. Diese Informationen sind nötig, um z.B. den Empfänger der gesendeten relevanten Informationen zu identifizieren. Allerdings sind sie häufig auch vermeidbar; insbesondere wenn Felder vorgesehen sind, die bei manchen Übertragungen nicht gesetzt bzw. beachtet werden müssen, wie z.B. Sequenznummern bei garantiertem *in-order* Empfang.

2.3 Der Idealfall

Im Idealfall müssen Transceiver von Stationen genau dann aktiv sein, wenn sie relevante Informationen senden oder empfangen. Demnach ist die Zeit, in welcher eine Station aktiv ist, im optimalen Fall proportional zur versendeten bzw. empfangenen Datenmenge. Dieses Minimum ist in der Realität nicht zu erreichen. Gründe hierfür sind die in Abschnitt 2.2 genannten Ursachen von Energieverschwendung. Zur Lösung dieser Probleme sind allgemeine Voraussetzungen zu schaffen, die in Abschnitt 3.1 diskutiert werden. Die dort beschriebenen Protokolle versuchen sich dem Ideal anzunähern, indem sie *Idle Listening* verringern, jedoch im Gegenzug die Kollisionswahrscheinlichkeit und/oder den Overhead durch Kontrollnachrichten in Kauf nehmen müssen.

3 MAC-Protokolle für Sensor-Netzwerke

Der folgende Abschnitt stellt Protokolle auf MAC-Ebene vor, die Möglichkeiten zum Sparen von Energie bieten. Allerdings erlaubt die Vielzahl der veröffentlichten Lösungsvorschläge keine umfassende Diskussion, sodass sich die Ausarbeitung auf oft referenzierte Verfahren und aktuelle Lösungsvorschläge beschränken muss.

3.1 Gemeinsamkeiten und Unterschiede

Alle Protokolle haben das identische Ziel, den Transceiver möglichst lange im Schlafmodus zu halten, um die Energieeffizienz zu erhöhen. Dabei versuchen alle Verfahren, Dienstgüteeigenschaften nur bis zu einem akzeptablen Maß zu verschlechtern.

Damit zwei benachbarte Stationen zu einem gemeinsamen Zeitpunkt aktiv sind, ist eine Signalisierung (siehe B-MAC in Abschnitt 3.7) oder eine Synchronisation notwendig. Die meisten Protokolle verwenden eine Synchronisation und das in Abbildung 2 dargestellte Schema. Die Abbildung zeigt die Unterteilung in Aktiv- und Schlaf-Phasen, wobei jede Aktiv-Phase mit einem Synchronisationsslot beginnt. Die periodische Resynchronisation vor der Datenübertragungsphase ist notwendig, um einerseits neue Knoten zu

Abbildung 2: Synchronisation in Duty-Cycling [31].

synchronisieren und andererseits Ungenauigkeiten durch unterschiedlichen Uhrenraten auszugleichen. Unterschiedliche Uhrenraten führen andernfalls dazu, dass die Uhren zweier Knoten beliebig auseinander driften.

Die Protokolle unterscheiden sich stark in ihren Annahmen über das Verkehrsmuster und -aufkommen, die Topologie und die Knotenanzahl. Während einige Protokolle speziell für typische Sensor-Netz-Topologien entwickelt wurden, in denen wenige Daten an eine Senke gesendet werden, versuchen andere Protokolle auch für generische Netzwerkstrukturen klassische Dienstgüteeigenschaften mit Energieeinsparungen zu verbinden. Dementsprechend ist ein quantitativer Vergleich wenig aussagekräftig.

Ein weiteres Unterscheidungskriterium ist die Reichtweite der Synchronisation. Einige der Protokolle setzen ein netzwerkweites Synchronisationsverfahren voraus [7], während andere Protokolle das Etablieren verschiedener Synchronisationen erlauben [31, 32]. Bei letzteren Verfahren werden Brückenknoten nötig, die bei mehreren Aktiv-Phasen aktiv sind und eine Kommunikation zwischen unterschiedlichen Synchronisations-Clustern ermöglichen.

3.2 S-MAC

Sensor MAC (S-MAC) wurde in [31] erstmals veröffentlicht und ist eines der ersten wettbewerbsbasierten MAC-Protokolle, das sich mit Duty Cycling in drahtlosen Netzwerken beschäftigt. S-MAC verfolgt das Ziel, den Energieverbrauch zu verringern und dennoch eine gute Skalierbarkeit und Kollisionsvermeidung zu unterstützen. Das Protokoll wurde für keine spezielle Topologie, wie z.B. ein Sensornetzwerk mit einer einzelnen Senke, optimiert, sondern geht von einer Kommunikation zwischen beliebigen Knoten als Peers aus. Dabei wird ebenfalls keine statische Infrastruktur vorausgesetzt, d.h. Knoten sind frei positioniert, können sich (beschränkt) bewegen und Knoten können dem Netzwerk beitreten oder es verlassen.

Im Gegensatz zu vielen anderen energieeffizienten MAC-Protokollen, setzt S-MAC kein bestehendes Synchronisationsverfahren voraus. Stattdessen führt S-MAC ein eigenes Protokoll ein, dass eine Multi-Hop-Tick-Synchronisation bietet, die jedoch keine netzwerkweite Synchronisation garantiert. Dadurch bilden sich virtuelle Cluster, die jeweils in sich synchronisiert sind und einem eigenen *Schedule* folgen, d.h. alle Knoten eines Clusters sind gleichzeitig aktiv. Daraus folgt, dass manche Knoten mehrere Aktiv-Phasen haben müssen, um eine Inter-Cluster-Kommunikation zu ermöglichen.
Die (Re-)Synchronisation findet zu Beginn jeder Aktiv-Phase entsprechend dem Schema in Abbildung 2 statt. Zur initialen Synchronisation unterscheidet S-MAC zwei Knotenarten: *Synchronizer* und *Follower*. Synchronizer sind Knoten, die ihren eigenen Schedule definieren und die Synchronisation starten. Ein Knoten wird zum Synchronizer, indem er nach dem Starten während eines vordefinierten Zeitintervalls keine fremden Synchronisationsnachrichten empfängt. Er wählt daraufhin eine Zeitdauer, die angibt, wann seine Schlaf-Phase beginnt und propagiert diese Zeit in einer SYNC Nachricht. Ein Knoten, der eine SYNC-Nachricht empfängt und noch keinen eigenen Schedule gewählt oder übernommen hat, wird als Follower bezeichnet. Er übernimmt den Schedule aus einer empfangenen SYNC-Nachricht und propagiert die SYNC-Nachricht nach einem zufälligen Backoff weiter. Dabei wird die Zeit in der SYNC-Nachricht um die Größe des zufälligen Backoffs verringert. Dadurch beginnen alle Knoten, die direkt oder indirekt eine SYNC-Nachricht eines Synchronizers empfangen, zum gleichen Zeitpunkt mit ihrer Schlaf-Phase. Somit beginnen sie auch zum gleichen Zeitpunkt mit der folgenden Aktiv-Phase, da die Schlaf-Phasen eine konstante und allen Stationen bekannte Dauer besitzen, die vorab bei der Installation festgelegt werden muss. Die Knoten, die eine SYNC-Nachricht empfangen, aber bereits einen Schedule übernommen haben, übernehmen zusätzlich den zweiten Schedule, um als Brückenknoten zwischen unterschiedlichen Synchronisations-Clustern zu agieren. Dazu verwalten sie eine *Schedule-Tabelle*, die die Schedules von allen benachbarten Knoten enthält, und wachen in den Aktiv-Phase von allen bekannten Schedules auf.
Zur Beibehaltung der Synchronisation ist zu Beginn jeder Aktiv-Phase ein Synchronisations-Slot reserviert, in dem periodisch SYNC-Nachrichten übertragen werden. Um Kollision zwischen parallelen Übertragungen zu vermeiden, werden SYNC-Nachrichten mit CSMA/CA und zufälligem Backoff übertragen, d.h. ein Sender wählt eine zufällige Zeit und beginnt seine Übertragung nur, wenn bis nach Ablauf der Zeit das Medium nicht belegt ist oder war. Die SYNC-Nachrichten enthalten ebenfalls den zum Sendezeitpunkt relativen Zeitpunkt, wann der Sender der SYNC-Nachricht die nächste Schlaf-Phase erreicht. So können sich neue Knoten zu existierenden Schedules synchronisieren. Das Synchronisationsintervall, d.h. das Intervall, in dem ein Knoten SYNC-Nachrichten sendet, kann größer sein als das Aktiv/Schlaf-Intervall, sodass ein Knoten nicht in jeder SYNC-Phase SYNC-Nachrichten überträgt. Dadurch soll der Wettbewerb und die Kollisionswahrscheinlichkeit von SYNC-Nachrichten verringert werden. Im Gegenzug erhöht sich allerdings die Eintrittszeit eines Knotens in einen existierenden Synchronisations-Cluster.

Damit sich zwei benachbarte virtuelle Cluster gegenseitig entdecken, müssen die Knoten regelmäßig für die komplette Dauer des Synchronisationsintervalls aktiv bleiben. In [31] wird dies als *Periodic Neighbor Discovery* bezeichnet. Abhängig von der Größe des Synchronisationsintervalls kostet *Periodic Neighbor Discovery* sehr viel Energie.

Zur Reduktion der in Abschnitt 2.2 beschriebenen Quellen von Energieverschwendung führt S-MAC vier Mechanismen ein:

1. Periodische Aktiv- und Schlaf-Phasen verringern die Energieverschwendung aufgrund von Idle Listening (siehe Abschnitt 2.2.1). Durch die beschriebene Synchronisation bilden sich virtuelle Cluster, deren Knoten gemeinsame Aktiv-Phasen haben, in welchen sie untereinander kommunizieren können. In den Schlaf-Phasen können Knoten ihren Transceiver abschalten und dadurch Energie einsparen. In den Veröffentlichungen von S-MAC ist die Größe der Aktiv-Phase so ausgelegt, dass pro Aktiv-Phase maximal ein Datenrahmen gesendet wird [31, 32], wobei die Übertragung selbst nicht in der Aktiv-Phase abgeschlossen sein muss. Dies hat bei einer Multi-Hop-Übertragung den Nachteil, dass mindestens genauso viele Aktiv-Phasen benötigt werden wie die zu überbrückende Hop-Anzahl beträgt, was zu einer enormen Erhöhung der Ende-zu-Ende-Verzögerung führt.

2. Zur Verringerung von Kollisionen verwendet S-MAC physisches und virtuelles *Carrier Sensing* (siehe Abschnitt 2.2.2). Physisches Carrier Sensing erfolgt durch Abtasten des Mediums durch den physikalischen Layer. Bei virtuellem Carrier Sensing verwaltet jede Station einen Network Allocation Vector (NAV), der angibt, ob das Medium frei ist. Der NAV wird gesetzt, wenn Rahmen empfangen werden, die nicht an den eigenen Knoten adressiert sind. Hierzu verwendet S-MAC bei Unicast-Übertragungen RTS/CTS-Kontrollrahmen, wie sie auch in WLAN 802.11 benutzt werden [12]. RTS und CTS Rahmen enthalten neben Empfänger und Sender ebenfalls eine Dauer, die angibt wie lange die folgende Datenübertragung benötigt. Der NAV wird beim Empfang von RTS/CTS-Rahmen auf diese Dauer gesetzt. Da RTS/CTS-Rahmen im gleichen Kanal gesendet werden wie Datenübertragungen, nennt man das Vorgehen auch 'In-channel'-Signalisierung. Die Länge der DATA-Phase ist so ausgelegt, dass eine RTS/CTS-Sequenz (ohne folgende Datenübertragung) abgeschlossen werden kann. Aufgrund eines empfangenen RTS-Rahmens weiß die Station, an die der Rahmen gerichtet ist, dass sie Empfänger der folgenden Übertragung ist, und wartet mit dem Schlafmodus bis der Datenrahmen vollständig empfangen ist.
   Virtuelles CSMA scheint auf den ersten Blick überflüssig zu sein, da die Dauer der Aktiv-Phase nur für die Übertragung eines einzelnen Datenrahmens ausgelegt ist. D.h. ein Knoten könnte nach dem Empfang eines fremden RTS auch in den Schlafmodus wechseln und erst zur nächsten Aktiv-Phase wieder aufwachen, ohne dabei den NAV zu setzen. Denn der NAV würde bis in die Schlaf-Phase hinein gesetzt bleiben, sodass bei Ablauf des NAV keine Station einen Übertragungsversuch unternehmen würde. Man kann nur vermuten, dass sich die Autoren die Möglichkeit offen halten, die Aktiv-Phase größer auszulegen, um mehrere Datenrahmen pro Aktiv-Phase zu ermöglichen. Diese Auslegung wird auch in [29] bestätigt, indem von anderen Autoren implizit angenommen wird, dass eine Station in einer Aktiv-Phase mehrere Datenrahmen senden/empfangen kann (siehe Abschnitt 3.3).
   Die Übertragung von Broadcast-Rahmen erfolgt ohne RTS/CTS-Rahmen und die Kollisionswahrscheinlichkeit kann dadurch nur durch physisches Carrier Sensing verringert werden.

3. Um Energieverschwendung durch Overhearing zu verringern, verwendet S-MAC die Informationen des NAVs von der In-channel-Signalisierung, um den Transceiver während der Übertragung zwischen anderen Knoten abzuschalten (siehe Abschnitt 2.2.3). Ein gesetzter NAV bedeutet, dass ein Knoten aktuell weder Rahmen senden, noch auf eingehende Rahmen antworten sollte, da er eine fremde Übertragung stören könnte. Deshalb kann ein Knoten den Transceiver in dieser Zeit ohne Beeinflussung der Performanz in einen Schlafmodus versetzen.

4. Angetrieben von dem Problem, dass die Verfälschungswahrscheinlichkeit von großen Rahmen in drahtlosen Netzwerken oft sehr hoch ist und die resultierenden Kosten für Wiederholungen ebenfalls hoch sind, werden große Rahmen in kleinere fragmentiert. Bei der separaten Übertragung der Fragmente als eigene Datenrahmen, kommt durch RTS/CTS-Rahmen ein großer Overhead durch Kontrollnachrichten hinzu. WLAN begegnet diesem Problem bereits, indem es Rahmen fragmentiert und als Bursts überträgt, d.h. nach einer RTS/CTS-Sequenz kommt eine Folge von Datenrahmen, die bei erfolgreichem Empfang einzeln mit einem ACK bestätigt werden. Mit *Message Passing* wird diese Idee zur Reduzierung von Overhead durch Kontrollnachrichten in S-MAC aufgegriffen (siehe Abschnitt 2.2.4). Im

Gegensatz zu WLAN wird in der RTS/CTS-Sequenz die Zeitdauer für die Übertragung aller Fragmente angegeben. Zusätzlich ist die Restdauer der kompletten Übertragung in allen Daten- und ACK-Rahmen angegeben. Dadurch können unbeteiligte Knoten, die Daten-/RTS-/CTS-/ACK-Rahmen empfangen, während der Übertragung des Bursts in den Schlafmodus wechseln. Bei dem Empfang eines verfälschten Rahmens muss ausschließlich das Fragment wiederholt übertragen werden. Diese Wiederholung geschieht sofort, wodurch sich die Gesamtdauer erhöht. Durch die Angabe von der Restlänge der Übertragungsdauer in Daten- und ACK-Rahmen, erkennen unbeteiligte Knoten die Verlängerung und wechseln erneut in den Schlafmodus.

Alle Mechanismen, bis auf die Vermeidung von Overhearing, erhöhen die Ende-zu-Ende-Verzögerung in Multi-Hop-Szenarien. Durch die beschriebenen periodischen Schlaf-Phasen kann nur 1 Hop pro Aktiv-Phase überbrückt werden, wodurch mindestens genauso viele Zyklen wie Hops notwendig sind. Aus diesem Grund wurde S-MAC in [32] um einen Mechanismus mit Namen *Adaptive Listening* erweitert. Adaptive Listening beschreibt, dass Knoten, die aufgrund der Overhearing-Vermeidung den Transceiver in den Schlafmodus gesetzt haben, außerhalb der Aktiv-Phase erneut für eine vorgegebene Dauer aufwachen. Im Detail gehen zunächst alle Knoten, die einen nicht für sie bestimmten RTS-/CTS-/Daten-/ACK-Rahmen empfangen, in den Schlafmodus und wachen erneut auf, wenn die in dem empfangenen Rahmen angegebene Dauer des angekündigten Datentransfers verstrichen ist. Dadurch kann der Empfänger der Datenübertragung den gerade empfangenen Rahmen direkt an den durch Adaptive Listening außerplanmäßig aufgewachten Knoten weitersenden, wodurch sich die Verzögerung verringert und der Datendurchsatz erhöht. Allerdings ist dies nur möglich, wenn die folgende Aktiv-Phase nicht beeinträchtigt wird, d.h. die Übertragung des weitergeleiteten Rahmens darf nicht in die nächste Aktiv-Phase hineinreichen. Durch Adaptive Listening wird ein Rahmen somit ausschließlich einen zusätzlichen Hop pro Duty Cycle weitergeleitet.

Der Ablauf von Adaptive Listening ist in Abbildung 3 dargestellt. Knoten A möchte einen Datenrahmen an Knoten C via Knoten B senden. Zu Beginn der Aktiv-Phase wartet er einen zufälligen Backoff ab und sendet anschließend einen RTS-Rahmen an Knoten B, der darauf mit einem CTS antwortet. Knoten C hört dieses CTS und geht für die Dauer der mit dem CTS angekündigten Übertragung zwischen A und B in den Schlafmodus. Während C im Schlafmodus ist, überträgt A den Datenrahmen an B. Im Idealfall wacht C genau dann wieder auf, wenn die Übertragung abgeschlossen ist, d.h. zu diesem Zeitpunkt sind Knoten B und C wach. Dadurch kann B die Daten im aktuellen Zyklus an C weiterleiten. [32] geht nicht darauf ein, ob der Rahmen nach einem erneuten Backoff weitergeleitet wird oder nur nach einem kurzen Interframe-Spacing. In der Grafik ist angenommen, dass auch vor der zweiten Übertragung in der Schlaf-Phase ein zusätzlicher Backoff gewartet wird.

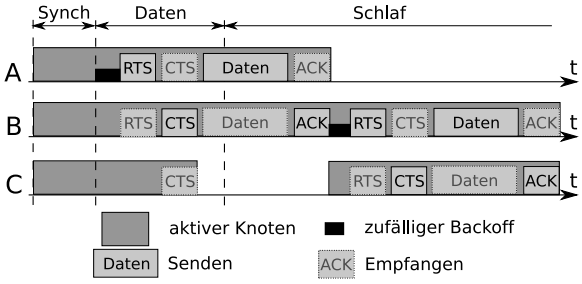

Abbildung 3: Erweiterung von S-MAC mit Adaptive Listening.

S-MAC wurde von den Autoren experimentell in einer Topologie mit 2 Hop Durchmesser evaluiert und der Energieverbrauch mit WLAN 802.11 [12] ohne Energieeinsparungsmodus verglichen [31]. Die Experimente zeigen, dass bei wenigen Rahmen der Energieverbrauch von S-MAC durch periodische Schlaf- und Aktiv-Phasen deutlich geringer ist als bei 802.11. Bei größeren Rahmenmengen kann sich S-MAC allerdings weniger energieeffizient als 802.11 verhalten. Gründe sind der zusätzliche Synchronisationsaufwand von S-MAC und die Tatsache, dass S-MAC länger braucht, um einen einzelnen Rahmen über mehrere Hops zu übertragen. Die Evaluationsergebnisse zeigen auch, dass der Energieverbrauch von S-MAC stark von dem Verkehrsaufkommen abhängt. Bei geringer Last kann der Transceiver häufiger in den Schlafmodus versetzt werden als bei hoher Last. Dies begründet sich damit, dass ein Knoten, der keinen Rahmen sendet oder empfängt, den Transceiver am Ende der Aktiv-Phase abschalten kann und Knoten durch Adaptive Listening seltener in der Schlaf-Phase erneut aufwachen. Dies wird auch durch Simulationen von Broadcast-Übertragungen in [27] bestätigt.

Trotz erkennbarer Verbesserungen im Vergleich zu existierenden Protokollen ohne Duty Cycling hat S-MAC einige Nachteile. In [18] und [7] werden neben der sehr hohen, zur Hop-Anzahl proportionalen Ende-zu-Ende-Verzögerung noch zwei weitere Probleme genannt. Ein Problem ist die fehlende Möglichkeit, den Duty

Cycle an aktuelle Bedürfnisse im Netzwerk anzupassen (z.B. eine Verlängerung der Aktiv-Phase bei hoher Last). Dadurch kann der erreichte Durchsatz geringer werden als der benötigte [29], obwohl die Ressourcen prinzipiell verfügbar wären. Adaptive Listening löst durch zusätzliches Aufwachen in der Schlafperiode dieses Problem nur begrenzt und hat außerdem den negativen Effekt, dass viele Knoten unnötigerweise erneut aufwachen, da sie nicht Empfänger einer Übertragung sind [7, 25]. Die Simulationen in [27] bestätigen die Schwäche durch Inflexibilität auch bei Broadcast-Übertragungen; insbesondere da Adaptive Listening durch fehlende RTS/CTS-Rahmen in diesem Fall nicht anwendbar ist. Die Inflexibilität führt hier zu einer Verringerung der PDR. Das zweite Problem ist die erhöhte Kollisionswahrscheinlichkeit in der Aktiv-Phase vor allem bei großen virtuellen Clustern, da mehrere Sender zur gleichen Zeit um das Medium konkurrieren. Des Weiteren sind in den Veröffentlichungen von S-MAC ([31, 32]) einige Fragen offen, die in anderen Arbeiten Platz für Interpretationen lassen. So wird z.B. in [29] die Länge der Daten-Phase so festgelegt, dass mehrere Datenrahmen empfangen/gesendet werden können. In [7] beschränkt sich die Größe der Daten-Phase hingegen auf den Versand/Empfang einer einzelnen RTS/CTS-Sequenz.

S-MAC setzt zwar keine netzwerkweite Synchronisation voraus, erhöht aber den Energieverbrauch bei mehreren virtuellen Clustern. Aus diesem Grund nehmen die Autoren auch an, dass sich in der Regel nur selten mehrere Schedules etablieren [31]. Diese Annahme spiegelt sich auch in den Evaluationen wider, da bei allen Evaluationen ein einziger virtueller Cluster gebildet wird. Zudem folgen im Zusammenhang mit mehreren etablierten Clustern weitere Probleme bei Überlagerung unterschiedlicher Aktiv-Phasen. Zum Beispiel können Knoten RTS/CTS-Rahmen, die vor ihrer Aktiv-Phase in einem anderen virtuellen Cluster gesendet wurden, nicht empfangen, stören aber unter Umständen durch eigene Übertragungen die angekündigte Datenübertragung. Solche Probleme könnten nur ausgeschlossen werden, wenn garantiert wird, dass die Aktiv-Phasen eines virtuellen Clusters 'tief' in den Schlaf-Phasen der anderen Cluster liegen.

3.3 T-MAC

Timeout-MAC (T-MAC) verspricht, flexibler und effizienter als S-MAC zu sein, sowohl bei Last, die über Zeit variiert, als auch bei Last, die netzwerkweit nicht homogen verteilt ist [29]. Die Grundidee ist, die Aktiv-Phase nicht vorab mit einer festen Länge zu definieren, sondern abhängig von der Last anzupassen.

Die Aktiv-Phase beginnt bei T-MAC – wie auch bei S-MAC – periodisch in einem festgelegten Intervall. Hierzu ist eine Synchronisation notwendig, die mit dem von S-MAC beschriebenen Protokoll durchgeführt wird. In ihren Beschreibungen gehen die Autoren allerdings nicht darauf ein, wann eine Resynchronisation stattfindet.

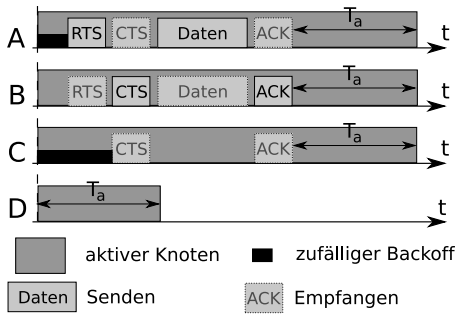

Abbildung 4: Grundidee von T-MAC [29].

T-MAC sendet ebenfalls die Datenrahmen nach einer erfolgreichen RTS/CTS-Sequenz und quittiert den erfolgreichen Empfang mit einem ACK-Rahmen. Aktiv-Phasen beginnen zwar in periodischen Abständen, enden aber nicht periodisch, d.h. sie haben eine variable Länge. Dazu definiert T-MAC eine Zeitspanne T_a (siehe Abbildung 4), die die Mindestlänge einer Aktiv-Phase festlegt. Wenn ein Knoten während der Zeitspanne T_a kein Ereignis auf dem Medium wahrnimmt, geht er in den Schlafmodus. In Abbildung 4 zeigt Knoten D dieses Verhalten als erster. Die Aktiv-Phase kann sich allerdings verlängern, wenn der Knoten selbst etwas sendet, ein RTS/CTS empfängt oder mittels Carrier Sensing eine fremde Übertragung erkennt. Bei Erkennen eines solchen Ereignisses bleibt ein Knoten zunächst aktiv und geht erst in den Schlafmodus, wenn nach Ende des Ereignisses das Medium während einer Zeitspanne von T_a nicht erneut als belegt erkannt wurde. Dadurch geht ein T-MAC-Knoten im Idealfall zu einem Zeitpunkt in den Schlafmodus, zu dem auch alle Nachbarknoten schlafen gehen. Vor dem Beginn einer RTS-Übertragung wartet eine Station (im Beispiel Knoten A) einen zufälligen Backoff aus einem Contention Window ab, welches T_a nicht überschreiten darf. Genauer betrachtet, muss in T_a nicht nur der maximal mögliche Backoff beachtet werden, sondern ebenfalls die Zeitspanne, die nötig ist, um einen Knoten in 2 Hop-Entfernung über das Ereignis zu informieren. D.h. T_a muss mindestens so groß gewählt werden, dass auch der Übertragungsbeginn eines CTS-Rahmens erkannt werden kann. Zusätzlich müssen Synchronisationsungenauigkeiten in T_a einfließen, damit Knoten mit einem Tick-Offset nicht zu früh schlafen gehen.

Wenn ein Knoten nach dem Senden eines RTS kein CTS erhält, kann dies mehrere Gründe haben: Der

RTS- bzw. CTS-Rahmen könnte verfälscht worden sein, der NAV des RTS-Empfängers könnte aufgrund eines fremden RTS/CTS-Rahmens gesetzt sein oder der RTS-Empfänger könnte bereits in den Schlafmodus gewechselt haben. Bei letzter Ursache könnte der Sender des RTS die Sendeversuche bis zur nächsten Aktiv-Phase einstellen. In [29] haben sich die Autoren für die energieineffizientere Variante entschieden, indem ein Knoten versucht den RTS-Rahmen dreimal zu senden, bevor er aufgibt und bis zur nächsten Aktiv-Phase wartet.
Bei der Vermeidung von Overhearing haben die Autoren ebenfalls eine energieineffizientere Alternative gewählt, da Knoten, die einen fremden RTS- oder CTS-Rahmen empfangen, nicht in den Schlafmodus wechseln (siehe Knoten C in Abbildung 4). Der Grund dafür ist, dass die Autoren durch Simulationen herausgefunden haben, dass der Durchsatz durch dieses unplanmäßige Schlafen reduziert wird, denn die Knoten überhören unter Umständen während der *Overhearing*-Schlafzeit weitere RTS/CTS-Rahmen. Dadurch können sie nach dem Wechsel in die Aktiv-Phase andere Übertragungen stören. Es wird demnach ein Kompromiss zwischen Overhearing bzw. Idle Listening und Kollisionen zu Ungunsten von Overhearing/Idle Listening gefällt. Die Autoren weisen aber darauf hin, dass das Ausschalten des Transceivers in solchen Fällen eine Option darstellt, wenn Verschlechterungen beim Durchsatz geduldet werden.
Das bisher beschriebene Protokoll hat das Problem, dass es durch die kurz gewählte Zeit T_a passieren kann, dass ein Knoten den Transceiver deaktiviert, obwohl ein benachbarter Knoten einen Sendewunsch zu dem Knoten hat. Das Problem lässt sich mit Abbildung 4 erläutern: C möchte eine Übertragung an D starten, verliert aber den Wettbewerb gegen A, der eine Übertragung an B startet. C erkennt seine Niederlage, indem er einen CTS-Rahmen von B empfängt. C kann einen neuen Übertragungsversuch erst starten, wenn die Übertragung zwischen A und B abgeschlossen ist, aber Empfangsknoten D ist zu diesem Zeitpunkt bereits im Schlafmodus, weil er das CTS nicht empfangen hat. In [29] wird dieses Problem als *Early Sleeping* Problem bezeichnet und es werden zwei Lösungen vorgeschlagen:

1. Mit einem *Future Request-to-Send*-Rahmen (FRTS) soll ein Empfangsknoten, in unserem Fall D, 'hingehalten' werden. Das Prinzip ist in Abbildung 5 dargestellt: Nachdem Knoten C den CTS-Rahmen von Knoten B empfangen hat und erkennt, dass er den Wettbewerb verloren hat, informiert er Knoten D mit einem FRTS-Rahmen, dass dieser in naher Zukunft Empfänger eines Datenrahmens wird. Da in dem empfangenen CTS-Rahmen die Dauer der Übertragung zwischen A und B enthalten ist, kann C in dem FRTS-Rahmen bereits den Zeitpunkt angeben, an dem er den neuen Übertragungsversuch zu Knoten D startet. D passt anschließend seinen Schedule an und wechselt nach Ablauf von T_a nicht in den Schlafmodus. Dadurch kann auch bei einer Multi-Hop-Übertragung maximal ein zusätzlicher Hop pro Aktiv-Phase überwunden werden [18].

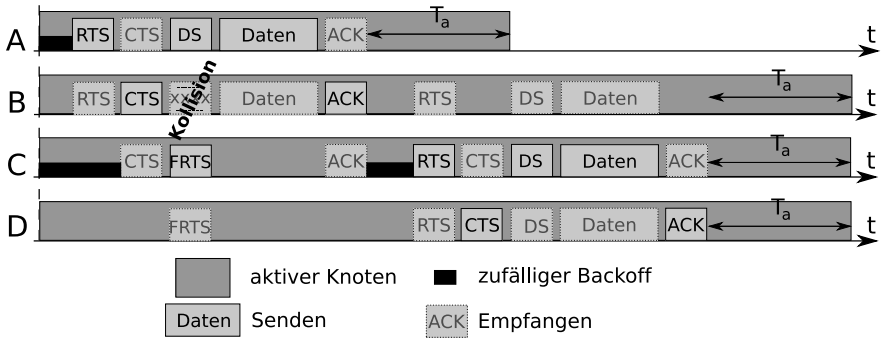

Abbildung 5: Future Request-to-Send als Lösung für das Early Sleeping Problem [29].

Abbildung 5 zeigt auch, dass der FRTS-Rahmen eine Verzögerung des Datenrahmens der ersten Übertragung nach sich zieht, da es zu einer Kollision zwischen Datenrahmen und FRTS-Rahmen kommen würde, wenn Knoten C die Datenübertragung direkt nach Empfang des CTS-Rahmens startet. Statt des Datenrahmens sendet A einen *Data-Send*-Rahmen (DS), der die Größe eines FRTS-Rahmens besitzt und verhindert, dass in der Zwischenzeit eine unbeteiligte Station um das Medium wirbt. Es ist aus diesem Grund gleichgültig, dass der FRTS-Rahmen mit dem DS-Rahmen kollidiert (im Beispiel bei Knoten B), denn der FRTS-Rahmen enthält nur Informationen für Knoten D, die auch nur dann von Interesse sind, wenn dieser zuvor kein RTS/CTS empfangen hat. Abbildung 5 lässt auch erkennen, dass die Dauer T_a verlängert werden muss, da D noch aktiv sein muss, wenn sowohl RTS als auch CTS

bereits vollständig gesendet wurden und die Übertragung des FRTS-Rahmen beginnt.
FRTS-Rahmen verringern zwar die Ende-zu-Ende-Verzögerung und erhöhen den Durchsatz, erhöhen dafür aber den Kontroll-Overhead und den Energieverbrauch. Wie bei der Vermeidung von Overhearing haben sich die Autoren von T-MAC für den Durchsatz und gegen den Energieverbrauch entschieden.
Obwohl in der Beschreibung von T-MAC das Prinzip nur auf CTS- und nicht auf RTS-Rahmen angewendet ist, lässt sich die Idee auch verfolgen, wenn ein Knoten anhand eines empfangenen und nicht an ihn adressierten RTS-Rahmens erkennt, dass er den Wettbewerb verloren hat.
Mit FRTS-Rahmen kann allerdings nicht garantiert werden, dass die Empfänger der Rahmen lange genug wach bleiben. Dies passiert z.B. im Falle einer Kollision zwischen zwei FRTS-Rahmen. In diesem Fall bleiben die Empfänger der FRTS-Rahmen zwar aufgrund der erkannten Kanalbelegung durch die Kollisionen länger aktiv, gehen aber direkt nach Verstreichen von T_a zu früh in den Schlafmodus. Aufgrund der Übertragung der FRTS-Rahmen ohne Mediumarbitrierung kann dieser Fall recht wahrscheinlich sein.

2. Die zweite Lösung ist nur indirekt eine Lösung für das *Early Sleeping* Problem und bringt eher den Vorteil, dass Knoten, die aktuell viele Rahmen puffern müssen, beim Mediumzugriff priorisiert werden. Die Lösung wird als *Full Buffer Priority* bezeichnet und erlaubt Knoten, die ein an sie adressiertes RTS empfangen, eine eigene Übertragung durch Senden eines neuen RTS-Rahmens zu starten. D.h. sie ziehen das Senden von eigenen Daten dem Empfangen fremder Daten vor. Die Idee wirkt absurd, da augenscheinlich nur selten relevante Informationen übertragen werden, wenn sich jeder Knoten so verhält. Deshalb ist die Lösung mehr für den 'Notfall' gedacht, damit sich ein Knoten von Daten befreien kann, bevor sein Puffer ausgeschöpft ist. Zusätzlich wurde es in T-MAC nur verwendet, wenn ein Knoten zuvor zweimal den Zugriff auf das Medium verloren hat.

T-MAC wurde sowohl simulativ mit S-MAC evaluiert als auch auf realer Hardware implementiert. Die Simulationsergebnisse zeigen, dass der Energieverbrauch von T-MAC und S-MAC – abhängig von gewählten Parametern wie Frame-Länge – bei homogener Last vergleichbar ist. Bei räumlich oder zeitlich ungleichmäßig verteilter Last, wie sie zum Beispiel in Baumtopologien von WSNs zu finden ist, schlägt T-MAC aufgrund seiner Möglichkeit zur dynamischen Anpassung der Aktiv-Phase S-MAC.
Die beschriebene Implementierung von T-MAC auf kleinen Sensorknoten[1] zeigte den Autoren die Notwendigkeit einer periodischen Resynchronisation, da in den Experimenten der Tick-Offset nach wenigen Minuten bereits so groß wurde, dass sich die Aktiv-Phasen mancher Knoten nicht mehr überlappt haben. Zusätzlich bestätigten Messungen des Energieverbrauchs, dass sich auch bei den verwendeten EYES-Knoten bereits durch Ausschalten des Transceivers sehr viel Energie einsparen lässt.

Der größte Nachteil von T-MAC zeigt sich in Netzen mit geringem Durchmesser, aber einer sehr hohen Knotendichte und einer regen Kommunikation zwischen sehr wenigen Knoten. Hier verhindert die dynamische Anpassung der Aktiv-Phase, dass Knoten in den Schlafmodus gehen, wenn sie eine Belegung des Mediums erkennen, obwohl unter Umständen nur sehr wenige Knoten in eine spätere Kommunikation verwickelt sind [18]. Bei Multi-Hop-Übertragungen erhöht sich die Verzögerung mit T-MAC ebenfalls wie bei S-MAC proportional zur Hop-Anzahl und pro Aktiv-Phase können in der Regel maximal 2 Hops zurückgelegt werden [25].

3.4 DMAC

Die Entwicklung von DMAC [18] wurde angetrieben durch das Fehlen einer Lösung für eine effiziente Multi-Hop-Übertragung von Rahmen trotz Duty Cycling. Die Autoren nennen dies *Data Forwarding Interruption*-Problem. DMAC geht von einer Baumtopologie aus, bei der Daten hauptsächlich von den Sensorknoten zu einer Senke hin gesendet werden, wie es häufig in WSNs zu finden ist. DMAC optimiert die Ende-zu-Ende-Verzögerung bei diesen senkenorientierten Übertragungen und geht davon aus, dass andere Kommunikationsmuster selten vorkommen und erlaubt dadurch eine Verschlechterung der Performanz für diese Fälle. Diese Fälle werden bei dem Design von DMAC nicht beachtet. Ebenfalls wie S-MAC und T-MAC ist DMAC ein wettbewerbsbasiertes MAC-Protokoll.

Zur Synchronisation setzt DMAC eine netzwerkweite Tick-Synchronisation voraus. Hierzu verwendet DMAC das Reference Broadcast Synchronisation Protokoll (RBS) [8], welches eine Multi-Hop-Synchronisation bietet. RBS kann eine hohe Synchronisationgenauigkeit erreichen, allerdings ohne dabei Garantien bezüglich der erreichten Genauigkeit oder benötigten Konvergenzzeit zu geben. Die Autoren gehen nicht auf den

[1] EYES Sensorknoten: http://www.eyes.eu.org/sensnet.htm

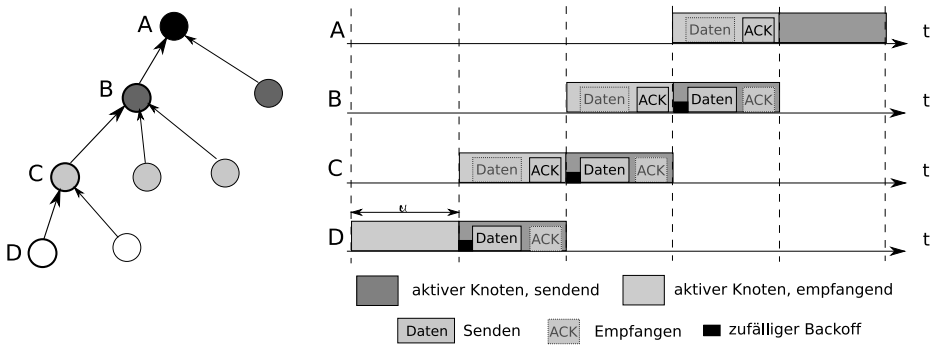

Abbildung 6: Grundschma von DMAC [18].

Overhead durch Synchronisation ein, oder wann (Re-)Synchronisation stattfindet. Allerdings ist die Konvergenzzeit von RBS in dichten Netzen sehr hoch, sodass die Autoren von einer Synchronisation ausgehen, wie sie unter Umständen mit RBS nicht erreicht werden kann. Auch die Wahl oder Bestimmung der Sender der Referenz-Broadcasts wird in [18] nicht thematisiert.

Die Grundidee von DMAC ist mit einer Kettenreaktion vergleichbar und ist in Abbildung 6 dargestellt. Die Abbildung zeigt die zeitliche Ausrichtung der Aktiv-Phasen in Abhängigkeit von der Position eines Knotens in der Baumtopologie. Die Aktiv-Phasen unterteilen sich in konstante, gleichgroße Listen- und Send-Slots, deren Dauer in [18] mit μ bezeichnet wird. Sie sind so angeordnet, dass sich der Listen-Slot von Knoten auf Ebene i mit den Send-Slots von Knoten auf Ebene $i+1$ überlappt. Dadurch kann im Idealfall ein Rahmen ohne langes Zwischenspeichern von einem Blattknoten zur Senke (Wurzel des Baumes) übertragen werden. Zusammen haben Send und Listen-Slot eine Dauer von 2 μ. Die Dauer μ hängt unter Anderem von der Größe der Nutzdaten ab, die von der Anwendung übertragen werden. Die Autoren von DMAC nehmen an, dass alle Datenrahmen die gleiche Größe haben. Andernfalls müsste eine Obergrenze angenommen werden, wodurch unter Umständen der Send- und Listen-Slot länger als nötig ist und Energie durch Idle Listening verloren geht.

Des Weiteren gehen die Autoren davon aus, dass die Rahmengröße in WSNs sehr gering ist. Damit begründen sie das Weglassen von RTS-/CTS-Rahmen, d.h. Datenrahmen werden in den Send-Slots nach einem zufälligen Backoff (aus einem Contention Windows) direkt gesendet und bei erfolgreichem Empfang mit einem ACK bestätigt.

Zusätzlich können Knoten während ihres Send-Slots Anfragen zur Hinzunahme einer weiteren Aktiv-Phase in der eigentlichen Schlaf-Phase mitsenden, um so bei hoher Last zusätzliche Rahmen in ihrer Schlaf-Phase zu übertragen. Die Vergrößerung des Duty Cycles wirkt sich dadurch nur auf dem Pfad zur Senke aus; andere Teilbäume sind davon nicht betroffen. Um den zusätzlichen Aktiv-Slot zu beantragen, hat DMAC ein *More Data* Flag in den Header von Daten- und ACK-Rahmen eingefügt. Falls ein Knoten mehr als einen Rahmen zu senden hat, setzt er in dem Datenrahmen, der in dem regulären Send-Slot übertragen wird, das Flag. Der Empfänger dieses Rahmens antwortet bei korrektem Empfang ebenfalls mit ACK, in dem das *More Data* Flag gesetzt ist, und leitet den Datenrahmen in seinem Send-Slot mit ebenfalls gesetztem *More Data* Flag weiter. Wenn ein Knoten einen Rahmen mit gesetztem *More Data* Flag empfangen hat, scheduled er eine zusätzliche Aktiv-Phase in der eigentlichen Schlaf-Phase. Diese zusätzliche Aktiv-Phase findet nicht direkt nach der ersten Aktiv-Phase statt. Der Grund hierfür ist in Abbildung 7 erkennbar: Eine direkte Aneinanderreihung würde dazu führen, dass sich die Send-Slots von Knoten auf Ebene $i-1$ mit dem Send-Slot von Knoten auf Ebene $i+1$ überlappen. Es käme zur Kollision bei dem Knoten auf Ebene i. Deshalb geht ein Knoten nach dem Send-Slot zunächst für die Dauer von 3 μ, d.h. für 1,5 Aktiv-Phasen, in den Schlafmodus und startet erst anschließend die zusätzliche Aktiv-Phase. Dies verringert die Ende-zu-Ende-Verzögerung von Datenrahmen eines Pfades.

Um die Ende-zu-Ende-Verzögerung aller Datenrahmen auf unterschiedlichen Pfaden zu verkleinern, wurde ein Mechanismus namens *Data Prediction* eingeführt. Mit *Data Prediction* kann ein Vaterknoten pro Aktiv-/Schlafzyklus mehr als einen Kindknoten bedienen, indem er auf Verdacht nach Bedienen (Empfangen und Weiterleiten von Datenrahmen) eines Knotens eine zusätzliche Aktiv-Phase in seiner eigentlichen Schlaf-Phase einführt. Die außerplanmäßige Aufwachzeit entspricht der gleichen Zeit, bei welcher der Knoten auch aufwachen würde, wenn er einen Rahmen mit gesetztem *More Data* Flag empfangen hätte. Dadurch weiß

ein Kindknoten, der zunächst den Wettbewerb gegen einen benachbarten Knoten mit gleichem Vaterknoten verloren hat, dass er einen zweiten Übertragungsversuch in der Schlaf-Phase unternehmen kann. Ein Vaterknoten scheduled diese zusätzlichen Aktiv-Phasen so lange, bis er in einer Aktiv-Phase keine Daten mehr empfängt. Da er nicht wissen kann, ob ein anderer Kindknoten einen Sendewunsch hat, geht er in jedem Fall einmal pro Zyklus in eine Aktiv-Phase und empfängt nichts. Hier kann er bereits direkt nach dem Listen-Slot in den Schlafmodus wechseln.

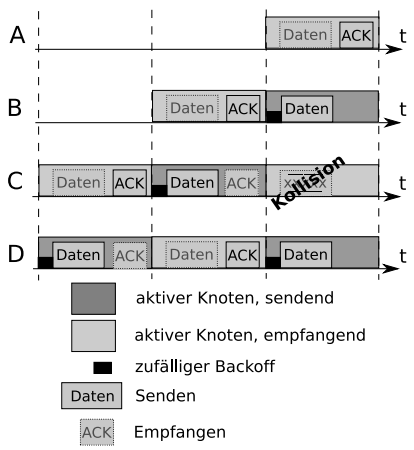

Abbildung 7: Kollisionen bei direkter Aneinanderreihung von Aktiv-Phasen.

Mit *More-To-Send* (MTS) Rahmen verbessert DMAC Szenarien mit Knoten, die in gegenseitiger Sendereichweite sind, aber unterschiedliche Vaterknoten haben. In solchen Szenarien schaltet ein Vaterknoten unter Umständen nach der ersten Aktiv-Phase zurück in den Schlafmodus, obwohl ein Kindknoten einen Sendewunsch hat und nur durch parallele Übertragungen in einem anderen Teilbaum davon abgehalten wurde, die Daten in dem regulären Send-Slot zu übertragen. MTS-Rahmen verlangen eine Verlängerung der Aktiv-Phase, da sie im Send-Slot zusätzlich zur regulären Übertragung von Datenrahmen gesendet werden. Sie sind sehr kurze Kontroll-Rahmen, die ein Flag enthalten, mit dem sie einem Vaterknoten mitteilen, ob er in seinem eigentlichen Schlaf-Zyklus regelmäßig in einem Intervall von 1,5 Aktiv-Phasen aufwachen soll oder nicht. Ein Kindknoten sendet einen MTS-Rahmen mit gesetztem Flag am Ende seines Send-Slots genau dann, wenn er zuvor den Wettbewerb um das Medium verloren hat, aber keinen ACK-Rahmen seines eigenen Vaterknotens empfangen hat. In diesem Fall wird angenommen, dass der Kindknoten den Wettbewerb gegen einen Knoten mit einem anderen Vaterknoten verloren hat. Durch den Erhalt eines MTS-Rahmens mit gesetztem Flag, wacht ein Vaterknoten periodisch in einem Intervall von 1,5 Aktiv-Phasen auf. Knoten, die einen MTS-Rahmen mit gesetztem Flag gesendet haben, erlösen den Vaterknoten von dem periodischen Aufwachen, wenn ihnen keine weiteren Rahmen vorliegen, indem sie einen MTS-Rahmen ohne gesetztes Flag an den Vaterknoten übertragen. Empfangene MTS-Rahmen (mit und ohne gesetztes Flag) werden jeweils an alle Vaterknoten bis zur Senke weitergesendet, sodass alle Knoten im Pfad außerplanmäßig aufwachen.
Die Autoren bleiben die Antwort auf die Frage schuldig, warum MTS-Rahmen mit nicht gesetztem Flag notwendig sind. Diese werden gesendet, wenn keine Daten mehr vorliegen, d.h. die Information könnte auch im *More Data* Flag des letzten Datenrahmens kodiert werden.

Ein Vorteil von DMAC liegt in der verringerten Kollisionswahrscheinlichkeit, denn zu einem beliebigen Zeitpunkt sind ausschließlich Knoten in Ebene i und $i+1$ aktiv, bzw. falls Knoten in zwei weiteren Ebenen j und $j+1$ aktiv sind, können im Regelfall Übertragungen zwischen $i/i+1$ bzw. $j/j+1$ stattfinden, ohne sich gegenseitig zu beeinflussen. Ein Wettbewerb findet daher in der Regel nur zwischen Knoten der gleichen Ebene statt.

Ein großes Problem von DMAC ist die implizite Annahme, dass sich alle Kindknoten eines gemeinsamen Vaterknotens gegenseitig hören können. Dies gilt in der Regel nicht immer, was zu dem Hidden Station Problem führt und in Abbildung 8 verdeutlicht ist. In dem Beispiel würden beide Kindknoten eine Übertragung starten, da sie sich gegenseitig nicht hören können. Es kommt zur Kollision bei dem gemeinsamen Vaterknoten, der daraufhin kein ACK schicken würde. Dies muss besonders kritisch betrachtet werden, da die Kindknoten in den folgenden Aktiv-Phasen immer wieder versuchen würden, den Rahmen zu senden bis er letztendlich verworfen wird. Falls das Contention-Window sehr groß gewählt wird, sodass zwei Datenrahmen nicht unbedingt kollidieren müssen (d.h. eine Station muss einen Backoff wählen, der größer als die Sendedauer der anderen Station ist), wäre ein Vorankommen eines Datenrahmens zwar prinzipiell möglich, jedoch abhängig von der Größe des Datenrahmens und der Länge des Contention

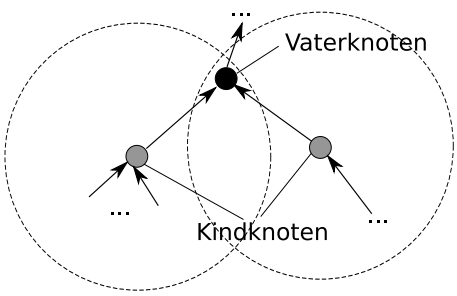

Abbildung 8: Kollisionen durch Hidden Station Szenario.

Windows nicht sehr wahrscheinlich. Außerdem würde bei großem Contention Window der Energieverbrauch durch Idle Listening steigen.

DMAC wurde in dem ns2-Netzwerksimulator gegen S-MAC und ein CSMA/CA basiertes MAC-Protokoll ohne Duty Cycling evaluiert. Unter anderem wurde in einem (praxisfernen) Multi-Hop-Szenario simuliert, in dem ein einzelnes Paket über eine unterschiedliche Anzahl an Hops übertragen wird. Ergebnisse aus diesen Simulationen zeigen, dass DMAC eine bessere Ende-zu-Ende-Verzögerung erreicht als S-MAC, aber vor allem bei 1-Hop Distanz CSMA/CA bei der Verzögerung unterlegen ist. Das begründet sich damit, dass der erste Knoten, der Daten zu senden hat, länger warten muss bis sein Send-Slot startet. Bei weiteren Hops kann DMAC allerdings seine Stärke ausspielen und die Verzögerung steigt nahezu identisch mit der Zunahme der Verzögerung bei CSMA/CA an. Der Energieverbrauch von DMAC ist in dem simulierten Szenario geringer als der Verbrauch von CSMA-CA und S-MAC, was laut Autoren daran liegt, dass bei DMAC nur die tatsächlichen Next-Hop Knoten aktiv sind und bei S-MAC auch Knoten, die evtl. (aus Netzwerksicht) mehr Hops entfernt sind, aber durch Adaptive Listening eine Kommunikation mithören und zusätzliche Aktiv-Phasen schedulen. Eine weitere Ursache liegt darin, dass S-MAC für die Ende-zu-Ende-Übertragung des Datenrahmens länger benötigt und dadurch Knoten deutlich häufiger in eine Aktiv-Phase kommen.
Die Ergebnisse der Simulationen sind allgemein sehr zu hinterfragen, da die Aktiv-Phase von S-MAC mit der gleichen Größe dimensioniert wurde, wie die Aktiv-Phase von DMAC. Allerdings muss in S-MAC – unter Vernachlässigung der Synchronisation – in der Aktiv-Phase nur eine RTS/CTS-Sequenz übertragen werden, in DMAC muss hingegen ein Daten- ACK-Rahmenpaar sowohl empfangen als auch gesendet werden, d.h. es muss in Relation zu S-MAC ein deutlich größerer Slot eingeplant werden.

Zusammenfassend lässt sich sagen, dass DMAC viele sinnvolle Ideen mit sich bringt, allerdings Annahmen trifft, die in der Realität nicht immer zutreffen müssen. Außerdem sind in der Beschreibung einige offene Fragen, z.B. warum das *More Send* Flag notwendig ist, wenn ein Vaterknoten wegen *Data Prediction* sowieso eine zusätzliche Aktiv-Phase einplant, falls er einen Rahmen empfangen hat. Hier lässt sich nur vermuten, dass einer Sequenz von Rahmen auf einem Pfad eine höhere Priorität zugeordnet wird und Nachbarknoten (mit gleichem Vaterknoten), die den Wettbewerb verloren haben, einen zweiten Sendeversuch erst starten, wenn sie einen Daten- oder ACK-Rahmen ohne gesetztes *More Send* Flag empfangen.
Ein weiterer Nachteil ist, dass die Optimierungen für Verzögerung und Energieverbrauch nur bei Baumtopologien Vorteile bringen, bei denen Nachrichten zur Senke hin fließen. Für generische Topologien oder bei wechselnden Verkehrsmustern, die eine Neuerstellung des Baumes verlangen, ist DMAC nicht geeignet [7].

3.5 RMAC

Das *Routing-Enhanced* MAC-Protokoll – kurz RMAC – entstand aus der Notwendigkeit, eine energieeffiziente Multi-Hop-Übertragung zu ermöglichen, ohne dabei spezielle Topologien vorauszusetzen wie es z.B. in DMAC getan wird (siehe Abschnitt 3.4) [7]. Neben der Verringerung der Ende-zu-Ende-Verzögerung hat RMAC die Verringerung des Wettbewerbs in der Aktiv-Phase und die Erhöhung des Durchsatzes zum Ziel.

Wie bei S-MAC und T-MAC wachen die Knoten bei RMAC zu einem gemeinsamen Zeitpunkt auf. RMAC setzt hierzu ein vorhandenes Synchronisationsprotokoll voraus, welches mindestens eine Tick-Synchronisation erreicht. Als Beispiele verweisen die Autoren auf das Reference Broadcast Synchronization Protokoll (RBS) [8] und Timing-Sync Protocol for Sensor Networks (TPSN) [9]. RMAC reserviert zu Beginn jeder Aktiv-Phase einen Synchronisations-Slot und folgt damit wie S-MAC dem Schema von Abbildung 2. Die Autoren gehen nicht auf die Größe der SYNC-Slots ein, was bei den verwiesenen Synchronisations-Protokollen zu einem Problem führt, da diese keine Konvergenzzeit garantieren.

Die Idee von RMAC ist in Abbildung 9 dargestellt und zeigt, wie Daten von Knoten A an Knoten D gesendet werden. RMAC basiert zunächst auf einem wettbewerbsbasierten Mediumzugriff in der Aktiv-Phase. Der Arbitrierungsvorgang entspricht dabei dem Verfahren von DCF in 802.11 [12], d.h. eine Station (im Beispiel Knoten A) wartet vor Sendebeginn ein zufälliges Backoff-Intervall aus einem Contention Window und ein Inter-Frame-Spacing DIFS ab und sendet nur, wenn während dieser Zeit keine Belegung des Mediums erkannt wurde.
Datenrahmen werden bei RMAC nicht in der Aktiv-Phase gesendet. Stattdessen finden in der Aktiv-Phase ausschließlich Übertragungen von PIONs (*Pioneer*-Rahmen) statt, die spätestens am Ende der Aktiv-Phase abgeschlossen sein müssen. PIONs ersetzen RTS/CTS-Rahmen und haben ähnliche Aufgaben. Im Gegensatz zu RTS-/CTS-Rahmen weisen sie nicht auf eine direkt folgende Übertragung hin, sondern auf eine Übertragung der Daten in der Schlaf-Phase. Mit PION-Rahmen wird zusätzlich Multi-Hop-Kommunikation beschleunigt. Dazu enthalten PIONs neben der Dauer der Datenübertragung, die Adressen von Sender und

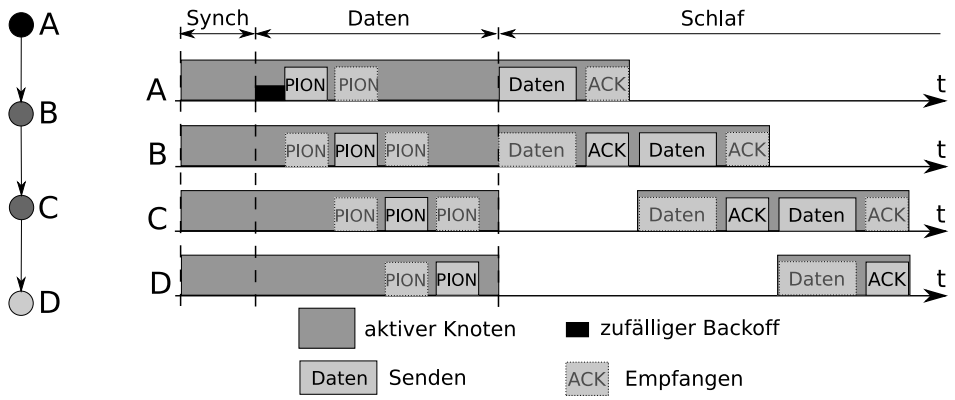

Abbildung 9: Grundidee von RMAC anhand einer Topologie mit 3 Hop Durchmesser [7].

direktem Next-Hop-Empfänger und zusätzlich die Adresse des Endknotens (im Beispiel Knoten D), welche ausschließlich bei einer Multi-Hop-Übertragung von Bedeutung ist. Dies ermöglicht Stationen, die auf der Route zu dem Endknoten liegen und ein PION empfangen (Knoten B und C), ein PION in der gleichen Aktiv-Phase weiterzuleiten, d.h. in unserem Beispiel kündigen B und C die Weiterleitung von Daten an, die zu diesem Zeitpunkt noch nicht empfangen wurden. Dabei ändern sie im PION die Adresse des Senders und bestimmen die Adresse des nächsten Knotens auf der Route zum Endknoten und tragen diese als Next-Hop-Empfänger im PION ein. PIONs dienen sowohl als RTS-Rahmen für den 'nächsten' Knoten, als auch als CTS-Rahmen für den 'vorherigen' Knoten in dem Multi-Hop-Pfad, sodass jeder (unbeteiligte) Empfänger eines PIONs über die Übertragung von Datenrahmen in der Schlaf-Phase informiert wird. Die Next-Hop-Information, die von jedem Knoten auf der Route bestimmt werden muss, liegt der MAC-Ebene in der Regel nicht vor und muss von dem Netzwerk-Layer erfragt werden.
Das Weiterleiten von PIONs erfolgt ohne Arbitrierung nach einem sehr kurzen Inter-Frame-Spacing. Dadurch ist die Verzögerung geringer, aber die Wahrscheinlichkeit einer Kollision von PIONs steigt. Wenn ein PION einen Endknoten erreicht hat, sendet dieser trotz des Fehlens eines nächsten Hops einen PION-Rahmen, um dem Vorgänger den Erhalt des PIONs zu bestätigen. In diesem Fall hat der PION die gleiche Funktion wie ein CTS.
Wie bereits erwähnt, schedulen PIONs die Übertragung von Datenrahmen in der Schlaf-Phase. Dadurch müssen PIONs Informationen mit sich führen, die angeben, wann die angekündigte Übertragung in der Schlaf-Phase stattfindet, sodass Empfänger und Sender gleichzeitig ihre Schlaf-Phase unterbrechen. Dazu wird ein zusätzliches Feld in den PION-Rahmen eingefügt, das die Hop-Anzahl angibt, die der PION in der aktuellen Aktiv-Phase bereits zurückgelegt hat, d.h. jeder Knoten inkrementiert bei Weiterleitung eines PIONs diese Zahl. Der Übertragungszeitpunkt der Datenrahmen berechnet sich daraufhin nach folgender Formel:

$$T_{wakeup}(n_{hops}) = T_{sleep} + (n_{hops} - 1) \cdot (D_{Data} + D_{SIFS} + D_{ACK} + D_{SIFS}),$$

wobei sich T_{sleep} anhand des Synchronisations-Ticks und der festen Dauer der Aktiv-Phase von allen Knoten ermitteln lässt. D.h. der Startzeitpunkt einer Übertragung beginnt relativ zu T_{sleep} (Beginn der Schlaf-Phase), zuzüglich eines Backoffs, der sich proportional zu der Hop-Anzahl n_{hops} berechnet und zusätzlich von der Dauer der angekündigten Datenübertragung (D_{Data}) inklusive Interframe Spacings ($2 \cdot D_{SIFS}$) und ACK-Rahmen (D_{ACK}) abhängt. Die Übertragung des ersten Rahmens beginnt direkt zu Beginn der Schlaf-Phase. Diese 'Pipelining'-Idee ist sehr ähnlich zu dem Pipelining in DMAC; mit dem Unterschied, dass die Pipeline in R-MAC on demand mit PIONs aufgebaut wird. Knoten bleiben demnach direkt zu Beginn der Schlaf-Phase aktiv (Knoten A und B) oder wachen genau dann wieder auf, nachdem die Sender, der von ihnen empfangenen PION-Rahmen, die Daten empfangen haben (Knoten C und D). PION-Übertragungen müssen vor Beginn der Schlaf-Phase abgeschlossen sein, um Kollisionen mit Datenrahmen zu verhindern. Ein Knoten darf einen PION daher nicht zu dem nächsten Hop weiterleiten, wenn die Zeit in der Aktiv-Phase nicht mehr ausreicht. Die maximale Anzahl an Hops, die ein Rahmen überwinden kann, ist demnach abhängig von der Länge der Aktiv-Phase.
Wie aus Abbildung 9 erkennbar ist, werden Datenrahmen ohne vorheriges Carrier Sensing zu dem geschedulten Zeitpunkt gesendet. Dies ist möglich, da PION-Rahmen die Funktion des virtuellen Carrier Sensing von

RTS/CTS-Rahmen übernehmen und erweitern. Im Detail heißt das, dass Knoten, die einen PION empfangen, den NAV zu mehreren Zeitpunkten setzen. Zunächst müssen Knoten den NAV direkt nach dem Empfang des PION setzen, da der Empfänger des PION den Empfang in den meisten Fällen mit einem neuen PION bestätigt. Zusätzlich muss der NAV auch in der Schlaf-Phase zu dem Zeitpunkt gesetzt werden, zu dem die angekündigte(n) Datenübertragung(en) stattfindet/stattfinden. Der Empfänger eines PION-Rahmens darf daher nur mit einem PION antworten, wenn der NAV aktuell und zu dem angestrebten Sendezeitpunkt nicht gesetzt ist. Dadurch ist im Regelfall, d.h. bei symmetrischen Links und einer geringen Dynamik in der Topologie, sichergestellt, dass keine Datenrahmen kollidieren.
Eine Kollision zweier PIONs hat zur Folge, dass die korrespondierenden Datenrahmen in der Schlaf-Phase maximal bis zu den Knoten gelangen, die als letzte Knoten im Pfad einen PION empfangen haben. Um Verlust und Verfälschung von Datenrahmen zu erkennen, wird jeder Empfang eines Datenrahmens mit einem ACK-Rahmen bestätigt (siehe Abbildung 9). Erhält ein Sender kein ACK, muss er davon ausgehen, dass die Daten nicht korrekt empfangen wurden und dass er sie neu übertragen muss. In diesem Fall kann es passieren, dass einige Stationen, die zuvor PIONs empfangen haben, umsonst ihre Schlaf-Phase unterbrechen, d.h. keine Datenrahmen empfangen [25]. Ein Knoten, der kein ACK als Antwort auf eine Datenübertragung erhalten hat, darf den Datenrahmen nicht in der aktuellen Schlaf-Phase erneut senden, da dadurch unter Umständen andere Übertragungen gestört werden. Stattdessen muss die Neuübertragung – wie nach einer Kollision zwischen PIONs – in der nächsten Aktiv-Phase mit einem PION neu angekündigt werden.

RMAC wurde in verschiedenen Topologien meist unter sehr geringer Last (z.B. 1 Paket alle 25 s) mit dem ns-2 Simulator [14] evaluiert. Um einen Vergleich zu existierenden Protokollen zu erhalten, wurden die Simulationen ebenfalls mit S-MAC durchgeführt. In den Topolgien mit großem Netzwerkdurchmesser zeigen die Ergebnisse erwartungsgemäß den Vorteil von RMAC, die Ende-zu-Ende-Verzögerung bei Multi-Hop-Übertragungen deutlich verringern zu können. Auch der Energieverbrauch von RMAC ist in diesen Topologien geringer als der Verbrauch von S-MAC. Die Autoren führen dies unter Anderem auf den geringeren Kontroll-Overhead in RMAC zurück, da nur ein PION-Rahmen anstelle eines RTS und eines CTS benötigt wird. Hauptgrund dürfte sein, dass RMAC weniger Aktiv-Phasen benötigt als S-MAC, um den Rahmen über mehrere Hops zu transportieren.

RMAC zeigt große Vorteile bei der Reduzierung der Ende-zu-Ende-Verzögerung bei Multi-Hop-Kommunikationen, wobei es in Single-Hop-Szenarien kaum Verbesserungen bringt. Ein weiterer – von den Autoren genannter – Vorteil ist die Eigenschaft, dass Rahmen bei Last schnell über mehrere Hops aus dem Überlastgebiet transportiert werden. Dieses Argument dürfte allerdings nur gelten, wenn sich die Last auf wenige Knoten verteilt. Bei einer hohen und verteilten Last ist hingegen zu erwarten, dass es zu vielen Kollisionen zwischen PION-Rahmen kommt; insbesondere da nur vor dem ersten PION eine Mediumarbitrierung durchgeführt wird. Obwohl die Autoren schreiben, dass RMAC auch funktioniert, wenn keine netzwerkweite Synchronisation besteht und sich mehrere Synchronisations-Cluster im Netz bilden, ist diese Aussage stark zu hinterfragen. Die Autoren schlagen für diesen Fall eine ähnliche Lösung wie in S-MAC vor, d.h. dass Knoten mehrere Schedules annehmen und Daten, die zwischen den Clustern ausgetauscht werden, puffern und in den jeweiligen Aktiv-Phasen mit PIONs ankündigen. Allerdings bedeuten mehrere Synchronisationen, dass sich Aktiv-/Schlaf-Phasen überlappen, wodurch Datenrahmen und PIONs kollidieren können. Dieser Vorschlag muss daher als sehr kritisch angesehen werden; vor allem da der Sender eines Datenrahmens den Übertragungszeitpunkt mit eigenen PIONs zuvor reserviert hat. Für die Simulationen gehen die Autoren von einer netzwerkweiten Synchronisation aus, die deutlich sinnvoller erscheint als Synchronisations-Cluster.

3.6 Fazit

Alle vier Protokolle – S-MAC [31, 32], T-MAC [29], DMAC [18] und RMAC [7] – haben Idle Listening als Hauptursache von Energieverschwendung ausgemacht. Zur Eindämmung von Idle Listening führen alle Protokolle Schlaf-Phasen ein, in denen die Knoten den Transceiver in einen Schlafmodus versetzen. Die letzteren drei Protokolle zeigen, dass die einfache Lösung von S-MAC, regelmäßige Schlaf-Phasen mit konstanter Länge einzuführen, zu inflexibel ist und die Ende-zu-Ende-Verzögerung in Multi-Hop-Szenarien enorm vergrößert, was den Anforderungen in manchen Sensor-Netzwerken nicht genügt. Aus diesem Grund haben sie Optimierungen eingeführt, die allerdings auf unterschiedlichen Voraussetzungen basieren, wie z.B. einer Baumtopologie in DMAC.

Tabelle 2 fasst die vier Protokolle qualitativ zusammen, kann allerdings nur einen groben Überblick geben, da viele Eigenschaften einer genaueren Betrachtung bedürften. Ein quantitativer Vergleich ist nicht sinnvoll, da Energieverbrauch und Dienstgütecharakteristiken von Topologie, Verkehrsaufkommen und dem verwen-

	Optimierung von Energieverbrauch (IL/K/O/KO)	Ende-zu-Ende-Verzögerung bei Multi-Hop-Übertragung	Synchronisation	Effiziente Broadcasts	strukturelle Komplexität	Merkmal
S-MAC	begrenzt / begrenzt / ja / begrenzt	sehr hoch ($\sim \#Hops$)	eigenes Verfahren (Multi-Hop, nicht netzwerkweit, keine Garantien)	begrenzt	mittel (virtuelle Cluster)	periodisch, statisch, synchrone Aktiv-Phasen
T-MAC	ja / begrenzt / nein / nein	sehr hoch ($\sim \#Hops$)	Verfahren von S-MAC	begrenzt	mittel (virtuelle Cluster)	Adaptive Länge der Aktiv-Phasen
DMAC	ja / nein / nein / begrenzt	gering, wenn Richtung Senke	RBS (Multi-Hop, keine Garantien)	nein	hoch (Baum-topologie)	Optimierung Source→ Sink Komm.
RMAC	ja / begrenzt / ja / ja	gering	RBS, TPSN (Multi-Hop, keine Garantien)	nein	nicht spezifiziert, abhängig von Synchronisa-tion	Optimierung Multi-Hop Routing

Tabelle 2: Qualitativer Vergleich der betrachteten Protokolle inkl. Bewertung bezüglich Vermeidung der in Abschnitt 2 genannten Quellen von vermeidbarem Energieverbrauch. (IL=Idle Listening, K=Kollisionen, O=Overhearing, KO=Kontrollnachrichten-Overhead)

deten Duty Cycle abhängen, die sich bei den Evaluationen stark unterscheiden. Bei den Evaluationen in den einzelnen Veröffentlichungen wurden größtenteils Szenarien verwendet, in denen sich das jeweils vorgestellte Protokoll günstig verhält.
Die Tabelle zeigt, dass alle Protokolle Kompromisse eingehen, um Ende-zu-Ende-Verzögerungen mit Energieverbrauch in Einklang zu bringen. So verringert RMAC die Ende-zu-Ende-Verzögerung, riskiert aber Kollisionen durch Senden von PIONs ohne vorheriges Carrier Sensing. Auch bei den Optimierungsmöglichkeiten müssen die Protokolle Kompromisse eingehen, wie z.B. DMAC, das den Overhead durch RTS/CTS-Kontrollrahmen verringert, dafür aber die Kollisionswahrscheinlichkeit in Hidden-Station-Szenarien erhöht. Zudem führt es neue Kontrollrahmen (MTS-Rahmen) ein, um bei ungünstiger Topologie, die Ende-zu-Ende-Verzögerung zu verringern.
Neben den Vergleichskriterien in der Tabelle gibt es noch weitere Kriterien. Zum Beispiel beschäftigt sich S-MAC als einziges Protokoll mit dem Problem, große Pakete, die in mehrere Rahmen fragmentiert werden müssen, energieeffizient zu versenden.

Tabelle 2 zeigt auch einige Schwächen der behandelten Protokolle. Obwohl alle Protokolle die Notwendigkeit einer Tick-Synchronisation erkennen, geht keines der Protokolle auf die Konvergenzzeit oder die Ungenauigkeiten in der Synchronisation ein, die Auswirkungen auf die Länge der Aktiv-Phase haben. Bei S-MAC und RMAC kann es dadurch passieren, dass die Länge des SYNC-Slots nicht ausreicht, um Nachbarknoten zu resynchronisieren, und folglich die Synchronisationsungenauigkeit nicht nur vom Synchronisationsintervall abhängt. Bei T-MAC und DMAC wird der Synchronisations-Overhead überhaupt nicht beachtet, sodass die Frage offen bleibt, wann die (Re-)Synchronisation stattfindet. Insbesondere in dem Schedule-Schema von DMAC ist die Antwort hierauf nicht offensichtlich.
Alle vier Protokolle setzen in ihren Simulationen und Experimenten voraus, dass Routen vorab bekannt sind. In realen Netzen ist dies nicht gegeben, sodass ein Routing-Protokoll benötigt wird, das auf dem jeweiligen MAC-Protokoll aufsetzt. Dies führt zu der Frage, wie effektiv Broadcasts bei den verschiedenen Protokollen verschickt werden können, da viele Routing Protokolle (wie AODV [21] oder DSDV [22]) auf Broadcasts basieren. Keines der vier behandelten Protokolle gibt Broadcasts eine gesonderte Bedeutung, wodurch diese in der Regel sehr ineffizient oder gar nicht übertragen werden können. Als Beispiel betrachten wir RMAC, wo ein Knoten einen Datenrahmen nur dann überträgt, wenn er ein PION gesendet und ein PION von dem Empfänger der angekündigten Daten empfangen hat. Dieses Schema funktioniert bei Broadcasts nicht, da ein Broadcast-Rahmen mehrere Empfänger hat und die Antwort-PIONs kollidieren würden (aus diesem Grund werden auch keine RTS/CTS-Rahmen bei Broadcasts verwendet). D.h. in RMAC können Broadcasts nicht nach dem gleichen Verfahren wie Unicasts versendet werden und es sind somit Erweiterungen notwendig.
In den betrachteten Veröffentlichungen wurden DMAC und RMAC ausschließlich simuliert und nicht auf realer Hardware implementiert. Diese rein simulativen Evaluationsergebnisse müssen daher mit Vorsicht betrachtet werden, da Simulationen die Realität nur ungenau widerspiegeln – wie unter Anderem in [17]

nachgewiesen wurde.

Das Wesentliche, was aus der detaillierten Betrachtung in den vorherigen Abschnitten und Tabelle 2 hervorgeht, ist einerseits die Folgerung, dass es das perfekte Duty-Cycling-Protokoll für generische Szenarien nicht geben kann. Andererseits sind Energieeinsparungen und Performanz im Sinne von Durchsatz/Verzögerung gegensätzliche Ziele. Je genauer das Wissen über Topologie, Nachrichtenmuster und benötigter Dienstgüte ist, desto mehr lässt sich ein Kompromiss finden, der das Protokoll für die Anwendung maßschneidert.

3.7 Weitere Ansätze

Die bisher genannten Protokolle stellen nur einen Bruchteil der veröffentlichten Lösungsvorschläge für Duty Cycling in Drahtlosnetzen dar. Dieser Abschnitt soll weitere Konzepte vorstellen, die die Vielfalt der Lösungen aufzeigen – ohne dabei sehr ins Detail zu gehen.

Real Time and Reliable MAC (RRMAC, [16]) kann als Erweiterung von 802.15.4 [13] für den speziellen Fall einer Baumtopologie angesehen werden. Ebenfalls wie 802.15.4 unterteilt RRMAC die Aktiv-Phase in einen Beacon-Slot, einen wettbewerbsbasierten Slot und einen wettbewerbsfreien Slot. Die einzelnen Slots werden weiter unterteilt und hierarchisch an Knoten entsprechend ihrer Position in der Baumtopologie zugewiesen, sodass in dem Beacon-Slot Beacons schnell von der Wurzel zu den Blättern und in dem wettbewerbsfreien Slot Datenrahmen schnell von den Blättern zu der Wurzel fließen. RRMAC funktioniert nur unter der Annahme, dass sich Knoten mit unterschiedlichen Vaterknoten bei parallelem Senden nicht gegenseitig stören.

Im Gegensatz zu den bisher diskutierten Protokollen kommt *Sparse Technology and Energy Management* (STEM, [24]) ohne ein Synchronisationsverfahren aus. Um eine Kommunikation zwischen zwei Nachbarknoten zu ermöglichen, wachen alle Knoten in regelmäßigen Abständen (unsynchronisiert) auf. Ein Knoten mit Sendewunsch muss vor der Übertragung einen Linkaufbau initialisieren. Der Linkaufbau besteht aus permanentem Senden von Beacon-Frames, bis letztendlich der Empfänger einen Beacon in dessen Aktiv-Phase empfangen kann und auf den Beacon antwortet. Anschließend können Daten übertragen werden. STEM geht von Transceiver-Hardware aus, die auf zwei Kanälen gleichzeitig senden und empfangen kann. Durch zwei Kanäle unterscheidet STEM zur Vermeidung von Interferenzen zwischen Aufwach-Kanal, der ausschließlich für Beacons verwendet wird, und Datenkanal, in dem reguläre Datenrahmen übertragen werden. Diese Lösung wird auch als Out-of-band-Signalisierung bezeichnet [15]. Der offensichtliche Nachteil von STEM ist der Energieverbrauch bei Sendeknoten, der sehr stark von dem Aufwachintervall der Knoten abhängt.

Berkeley MAC (B-MAC, [23]) kommt ebenfalls ganz ohne Synchronisation aus und reduziert damit einen großen Teil des Overheads. Stattdessen führt B-MAC ein Präambel-Sampling-Schema ein, d.h. bevor ein Knoten einen Datenrahmen überträgt, belegt er das Medium solange mit einem Dummy-Rahmen, um jedem Knoten das Erkennen des Sendewunsches zu ermöglichen. B-MAC ist somit sehr ähnlich zu STEM, wobei die Signalisierung bei B-MAC *in-channel* geschieht. Damit haben andere Knoten die Chance, den Sendewunsch eines Knotens zu erkennen, indem sie regelmäßig aufwachen und das Medium auf Belegung prüfen. Die Länge der Aufwachzeit lässt sich ebenso wie das Prüfintervall durch die Middleware konfigurieren, wodurch B-MAC dynamische Anpassungen ermöglicht. B-MAC selbst ist ein kleingehaltenes Protokoll, dass keine Unterstützung für Fragmentierung oder RTS/CTS-Rahmen zur Lösung des Hidden Stations-Problem bietet. Durch den konfigurierbaren Charakter lassen sich solche Erweiterungen aber auf B-MAC aufsetzen. Ein großer Nachteil ist die sehr große Ende-zu-Ende-Verzögerung bei Multi-Hop-Übertragungen, da jeder Knoten eines Pfades nach Erhalt des Rahmens warten muss, bis der nächste Hop die Aktiv-Phase betritt.

Das in [15] beschriebene *Wakeup Scheduling in Wireless Ad-Hoc Networks* stellt eine Generalisierung von DMAC dar (siehe Abschnitt 3.4). Das Protokoll benötigt ebenfalls eine Tick-Synchronisation und wurde für die Anwendung in einer Baumtopologie entwickelt. In Gegensatz zu DMAC, bei dem der Datenfluss von den Blättern zur Senke optimiert ist, behandeln die Autoren in [15] unterschiedliche Aufwach-Muster und beschäftigen sich mit der maximalen Ende-zu-Ende-Verzögerung in beiden Richtungen, d.h. zusätzlich von der Senke zu den Blättern. Die Analyse geht nicht auf die Art des Mediumzugriffs ein, sodass mögliche zusätzliche Verzögerungen durch Wettbewerb in den einzelnen Baumebenen nicht beachtet werden. In [15] wird mit der *Multi-Parent*-Methode zusätzlich ein Verfahren vorgeschlagen, wie bei mehreren redundanten Routen von Knoten zur Basisstation, d.h. bei Auflösen der strikten Baumtopologie, Energie eingespart bzw. die Verzögerung verringert werden kann, indem einem Knoten mehrere Vaterknoten zugeordnet werden und die Aufwachzeitpunkte der Vaterknoten durch ein Offset gegeneinander verschoben werden.

In [19] beschäftigen sich die Autoren (unter Anderem ist einer der Autoren ebenfalls Autor von DMAC [18]) damit, wie die Aufwachzeitpunkte in beliebigen Netzwerktopologien verschoben werden müssen, um Verzögerungen und Energieverbrauch zu minimieren. Die Autoren betrachten ein Duty Cycling, bei dem

ein Frame in k Slots unterteilt ist und jedem Knoten genau ein regulärer Aufwachslot aus $\{0, \ldots, k-1\}$ zugeordnet ist. Sie beweisen, dass es NP-schwierig ist, eine Slotzuweisung in einer beliebigen Topologie zu finden, sodass die Verzögerung aller kürzesten Pfade minimal ist. D.h. es gibt keinen effizienten Algorithmus, der eine solche Slotzuweisung durchführt. Die Autoren zeigen, dass bei speziellen Topologien diese Aussage nicht unmittelbar gelten muss, indem sie für Baum- und Ringtopologien eine entsprechende Slot-Zuweisung angeben. Zusätzlich werden für beliebige Topologien Heuristiken beschrieben.

Demand Wakeup MAC (DW-MAC, [25]) stellt eine Erweiterung von RMAC dar, das in Abschnitt 3.5 diskutiert wurde. Es optimiert – wie RMAC – die Multi-Hop-Übertragung in beliebigen Topologien, indem in der Aktiv-Phase Datenübertragungen mittels wettbewerbsbasiert versendeten Schedule-Rahmen (SCH) für die Schlaf-Phase geschedult werden. Die in DW-MAC eingeführten SCH-Rahmen entsprechen den PION-Rahmen in RMAC und übernehmen ebenfalls die Rolle von RTS/CTS-Rahmen. Der Unterschied ist, dass SCH-Rahmen nicht die Hop-Anzahl mitführen, die sie in der Aktiv-Phase bereits zurückgelegt haben. Stattdessen wird ein 1:1-Mapping verwendet, um den Sendezeitpunkt eines SCH-Rahmens in der Aktiv-Phase auf den Sendezeitpunkt des Datenrahmens in der Schlaf-Phase abzubilden. Die Beschreibung von DW-MAC in [25] ergänzt das in RMAC [7] fehlende Verfahren, um Broadcasts zu übertragen. Im Detail heißt dies, dass Broadcasts wie Unicasts mit SCH-Rahmen in der Aktiv-Phase angekündigt werden, die aber von den Empfängern nicht direkt, ohne Backoff, mit einem SCH-Rahmen bestätigt werden. Allerdings kann sich jeder Empfänger eines solchen SCH-Rahmens bereits in der aktuellen Aktiv-Phase neu für das Medium bewerben, um den in der Schlaf-Phase folgenden Datenrahmen in dem gleichen Zyklus weitertransportieren zu können. Um Broadcasts effektiver zu verbreiten, schlagen die Autoren ein zusätzliches Feld namens *Immediate Forwarder* in dem SCH-Rahmen vor, welches die Adresse eines Nachbarknotens angibt, der den SCH-Rahmen direkt und ohne erneuten Wettbewerb weitersendet. Dadurch können sich Broadcasts auf einem Pfad schneller ausbreiten.

4 Zusammenfassung

Diese Ausarbeitung beschäftigt sich mit dem Thema Duty-Cycling in drahtlosen Netzwerken mit dem Hauptaugenmerk auf WSNs. Ein Duty Cycling Protokoll hat zunächst die Aufgabe, periodische Aktiv- und Schlafphasen so zu synchronisieren, dass zwei Nachbarknoten miteinander kommunizieren können. Doch in vielen WSNs ist ein gemeinsamer Aufwachzeitpunkt nicht ausreichend, da Applikationen zusätzliche Anforderungen an Dienstgüteeigenschaften wie Ende-zu-Ende-Verzögerungen haben. Die große Herausforderung eines MAC-Protokolls mit Duty Cycling ist daher einerseits den Energieverbrauch zu minimieren, aber andererseits auch bei sporadisch auftretenden Ereignissen Informationen schnell und zuverlässig an ihren Bestimmungsort zu transportieren. In manchen Anwendungsfällen, wie z.B. Feuermelde-Systemen, wird diese Aufgabe umso schwieriger, da Ereignisse selten, aber korreliert, von mehrere Sensorknoten erkannt werden.

In der vorliegenden Ausarbeitung wurden zunächst die Ursachen von vermeidbarem Energieverbrauch betrachtet. Dabei identifizierten alle betrachteten Protokolle Idle Listening als größte Ursache. Anschließend wurden vier Protokolle im Detail diskutiert. S-MAC kann – als eines der ersten Protokolle, dass sich dem Thema Duty Cycling in Drahtlosnetzen annimmt – den Energieverbrauch bereits stark verringern, hat allerdings weiteres Potential bei Energieeinsparungen, ist zu inflexibel und zeigt Schwächen bei Multi-Hop-Übertragungen. T-MAC verspricht flexibler zu sein als S-MAC, hat allerdings auch Nachteile in Multi-Hop-Szenarien. In DMAC wird der Spezialfall von Baumtopologien betrachtet, wie sie in WSNs häufig zu finden sind. DMAC ordnet die Aktiv-Phasen so an, dass Informationen mit geringer Verzögerung von den Blättern zur Senke fließen, ist aber bei anderen Verkehrsmustern performanzschwach und kann in anderen Topologien nicht eingesetzt werden. Mit RMAC wurde zum Schluss ein Protokoll vorgestellt, dass Multi-Hop-Übertragungen in beliebigen Topologien optimiert. In den Evaluationen zeigt RMAC bereits sehr gute Resultate. Es ist aber zu erwarten, dass es in dichten Netzen mit hohem Nachrichtenaufkommen mit vielen Kollisionen zu kämpfen hat.

Anschließend wurden in der Ausarbeitung weitere Protokolle in einer Zusammenfassung behandelt. Die Vielzahl der teilweise sehr unterschiedlichen Lösungsvorschläge sollte dabei zeigen, dass es das perfekte Protokoll für Duty Cycling nicht geben kann, da zu viele Faktoren wie Topologie, Verkehrsmuster, etc. eine Rolle spielen.

Die meisten diskutierten Protokolle benutzen Synchronisations-Protokolle, die keine Garantien bezüglich erreichter Genauigkeit oder benötigter Konvergenzzeit geben. Insbesondere in sehr dichten Sensor-Netzwerken ist die Konvergenzzeit der verwendeten bzw. referenzierten Synchronisations-Protokollen sehr

hoch. Diese Tatsache könnte bei manchen betrachteten Duty Cycling Protokollen zu einem Problem werden und es müsste untersucht werden, wie sich Synchronisationsungenauigkeiten und unbegrenzte Konvergenzzeit auf das Verhalten auswirkt.
In den Evaluationen werden bei allen Veröffentlichungen existierende und statische Routen vorausgesetzt. Der Overhead durch Routing-Protokolle wird vollständig ignoriert. Viele Routing-Protokolle für Drahtlosnetze basieren auf Broadcasts, sodass ein Duty Cycling Protokoll eine effiziente Übertragung von Broadcasts ermöglichen sollte. Auch andere Applikationen, wie z.B. Sprachanwendungen, basieren auf Broadcasts und haben Anforderungen an das verwendete MAC-Protokoll [27]. Allerdings geht – bis auf DW-MAC – keines der Protokolle auf die Übertragung von Broadcasts ein.

Literatur

[1] Ian F. Akyildiz, Deborah Estrin, David E. Culler, and Mani B. Srivastava, editors. *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys 2003, Los Angeles, California, USA, November 5-7, 2003*. ACM, 2003.

[2] Lan F. Akyildiz, Welljan Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine, August 2002*, 2002.

[3] Giuseppe Anastasi, A. Falchi, Andrea Passarella, Marco Conti, and Enrico Gregori. Performance measurements of motes sensor networks. In Simonetta Balsamo, Carla-Fabiana Chiasserini, and Lorenzo Donatiello, editors, *MSWiM*, pages 174–181. ACM, 2004.

[4] ATmega128. Datenblatt atmel atmega128. `http://www.atmel.com/atmel/acrobat/doc2467.pdf`.

[5] Vaduvur Bharghavan, Alan J. Demers, Scott Shenker, and Lixia Zhang. MACAW: A media access protocol for wireless lan's. In *SIGCOMM*, pages 212–225, 1994.

[6] Crossbow Technology Inc. Datenblatt micaz. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[7] Shu Du, Amit Kumar Saha, and David B. Johnson. RMAC: A routing-enhanced duty-cycle MAC protocol for wireless sensor networks. In *INFOCOM*, pages 1478–1486. IEEE, 2007.

[8] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. In *OSDI*. Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, USA, 2002.

[9] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In Akyildiz et al. [1], pages 138–149.

[10] Jaap C. Haartsen, Ericsson Radio, and Systems B. V. The bluetooth radio system. *IEEE Personal Communications*, 7:28–36, 2000.

[11] Gertjan P. Halkes, Tijs van Dam, and Koen Langendoen. Comparing energy-saving MAC protocols for wireless sensor networks. *MONET*, 10(5):783–791, 2005.

[12] IEEE Std. 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Computer Society, 1999.

[13] IEEE Std. 802.15.4. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE Computer Society, 2003.

[14] USC Information Sciences Institute. The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[15] Abtin Keshavarzian, Huang Lee, and Lakshmi Venkatraman. Wakeup scheduling in wireless sensor networks. In Sergio Palazzo, Marco Conti, and Raghupathy Sivakumar, editors, *MobiHoc*, pages 322–333. ACM, 2006.

[16] Jungsook Kim, Jaehan Lim, Christopher Pelczar, and Byungtae Jang. RRMAC: A sensor network mac for real time and reliable packet transmission. Algarve, Portugal, april 2008. 12th Annual IEEE International Symposium on Consumer Electronics, IEEE.

[17] David Kotz, Calvin Newport, and Chip Elliott. The mistaken axioms of wireless-network research. Technical report, Dartmouth College, July 2003.

[18] Gang Lu, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. *Parallel and Distributed Processing Symposium, International*, 13:224a, 2004.

[19] Gang Lu, Narayanan Sadagopan, Bhaskar Krishnamachari, and Ashish Goel. Delay efficient sleep scheduling in wireless sensor networks. In *INFOCOM*, pages 2470–2481. IEEE, 2005.

[20] Tolga Numanoglu, Bulent Tavli, and Wendi Rabiner Heinzelman. Energy efficiency and error resilience in coordinated and non-coordinated medium access control protocols. *Computer Communications*, 29(17):3493–3506, 2006.

[21] Charles E. Perkins and Elizabeth M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *WMCSA*, pages 90–100. IEEE Computer Society, 1999.

[22] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *SIGCOMM*, pages 234–244, 1994.

[23] Joseph Polastre, Jason L. Hill, and David E. Culler. Versatile low power media access for wireless sensor networks. In John A. Stankovic, Anish Arora, and Ramesh Govindan, editors, *SenSys*, pages 95–107. ACM, 2004.

[24] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani B. Srivastava. Topology management for sensor networks: exploiting latency and density. In *MobiHoc*, pages 135–145. ACM, 2002.

[25] Yanjun Sun, Shu Du, Omer Gurewitz, and David B. Johnson. DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks. In Xiaohua Jia, Ness B. Shroff, and Peng-Jun Wan, editors, *MobiHoc*, pages 53–62. ACM, 2008.

[26] TAOS. Datenblatt TSL250R Lichtsensor. `http://www.goblack.de/desy/digitalt/sensoren/tsl-250/tsl250r.pdf`.

[27] Bulent Tavli and Wendi B. Heinzelman. QoS and energy efficiency in network wide broadcasting: A MAC layer perspective. *Computer Communications*, 30(18):3705–3720, 2007.

[28] TexasInstruments. Datenblatt cc2420. `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`.

[29] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In Akyildiz et al. [1], pages 171–180.

[30] Alec Woo and David E. Culler. A transmission control scheme for media access in sensor networks. In *MOBICOM*, pages 221–235, 2001.

[31] Wei Ye, John S. Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM*, 2002.

[32] Wei Ye, John S. Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.