Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

Abstufbare Authentizität und Non-Repudiation bei aggregierendem Datentransport im WSN

Diplomarbeit am Institut für Telematik
Prof. Dr. M. Zitterbart
Fakultät für Informatik
Universität Karlsruhe (TH)

von

cand. inform.
Joachim Wilke

Betreuer:
Prof. Dr. M. Zitterbart
Dr. Erik-Oliver Blaß

Tag der Anmeldung: 5. März 2007
Tag der Abgabe: 4. September 2007

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 04. September 2007

Menschen mit einer neuen Idee gelten so lange als Spinner,
bis sich die Sache durchgesetzt hat.

— Samuel Langhorne Clemens (Mark Twain)

Zusammenfassung

Protokolle für drahtlose Sensornetze erfordern besondere Rücksichtnahme auf beschränkte Ressourcen und erweiterte Angreifermodelle.

Einerseits steht den Sensorknoten nur beschränkte Energie und Rechenleistung zur Verfügung. Deshalb wird in vielen Szenarien auf Datenaggregation zurückgegriffen, um die Menge der zu übertragenden Informationen zu reduzieren und damit Energie zu sparen.

Andererseits sind die Knoten Angreifern ausgesetzt, die direkten physischen Zugriff erlangen können. Um dem Netzwerknutzer trotzdem zuverlässige und authentische Daten zu liefern, ist der Einsatz von Kryptographie unerlässlich.

Diese Arbeit zeigt auf, dass Authentizität und Aggregation gegensätzliche Anforderungen sind. Durch die Verwendung von Aggregation werden Daten so verändert, dass Authentizität nicht mehr sichergestellt werden kann.

Sie baut auf ESAWN auf, einem Protokoll zur authentischen Aggregation. ESAWN bietet durch Relaxieren der strengen Authentizitäts- und Aggregationsanforderungen einen Tradeoff zwischen Authentizität und Energiebedarf. ESAWN wird analysiert und die Nachteile des Protokolls werden aufgezeigt.

Anschließend werden zwei neue Protokolle, ESAWN-2 und ESAWN-NR, vorgestellt, die diese Lücken schließen und zusätzliche Funktionalität, wie zum Beispiel Non-Repudiation, bieten. Dies wird durch Mehrheitsentscheide über zusätzlich berechnete Aggregate und gegenseitiges Protokollieren der versandten Daten erreicht.

Die neuen Protokolle werden hinsichtlich ihres Energie- und Speicherbedarfs evaluiert, bewertet und mit den Ergebnissen von ESAWN verglichen. Alle Protokolle liegen bei Speicher- und Energieverbrauch pro Knoten in $O(1)$ bezogen auf die Netzwerkgröße.

Die abschließende Implementierung der Protokolle in TinyOS zeigt die praktische Realisierbarkeit der Protokolle und eignet sich zu Test- und Demonstrationszwecken.

Inhaltsverzeichnis

1. Einleitung

Die technische Entwicklung der vergangenen Jahrzehnte im Bereich der informationsverarbeitenden Technologien eröffnete für die Computertechnik immer wieder neue Anwendungsgebiete. Neben dem Wunsch nach stets größerer Rechenleistung, bestimmt eine fortschreitende Miniaturisierung den derzeitigen Trend.

Ein Teil dieses Trends besteht in der Kombination von Sensoren und Funktechnik, sowie Prozessoren, Arbeits- und Energiespeichern, zu so genannten Sensorknoten. Diese hochintegrierten und völlig autarken Kleingeräte sind in der Lage, spontan Netzwerke zu formen. Sie erfassen ihre Umwelt sensorisch und verarbeiten oder leiten Messdaten weiter. Während derzeit erst Sensorknoten von wenigen Kubikzentimetern Größe existieren, sprechen Visionäre schon vom so genannten „Smart Dust“ [23; 37], also tausenden Sensorknoten von der Größe eines Staubkorns, die sich unauffällig in unsere gewohnte Umwelt einfügen. Daraus ergeben sich völlig neue Möglichkeiten, etwa Umwelt- und Gebäudeüberwachung [32], medizinische [38] und militärische Applikationen [31; 28] und vieles mehr.

Doch auch an den bereits verfügbaren Systemen lässt sich die Herausforderung erkennen, die mit der Entwicklung von Softwaresystemen für solche Sensornetze („Wireless Sensor Networks“ – WSN) verbunden ist. Durch die beschränkten Energie- und Rechenressourcen lassen sich Konzepte und Protokolle aus der Welt von Internet und Hochleistungsrechnersystemen nicht unverändert übernehmen. Speziell auf Sensornetze und Anwendungsszenario zugeschnittene Konzepte sind gefragt. Sie müssen effizient mit den zur Verfügung stehenden Ressourcen umgehen und außerdem neue Sicherheitsanforderungen erfüllen, da Sensorknoten oft dem physikalischen Zugriff unautorisierter Personen ausgesetzt sind.

Energie ist die Ressource, die bei Sensorknoten mit am stärksten eingeschränkt ist. Einer der größten Energieverbraucher ist die drahtlose Kommunikationsschnittstelle des Gerätes [29; 40]. Um den Kommunikationsaufwand im Netzwerk zu reduzieren, und damit Energie zu sparen, wird häufig Datenaggregation [15; 19] eingesetzt. Datenaggregation kann immer dann eingesetzt werden, wenn der Anwender nicht an den detaillierten Messwerten eines jeden Sensorknotens interessiert ist, sondern Approximationen, Mittelwerte, Maxima oder andere Zusammenfassungen der Daten

ausreichen. Dazu wendet ein Sensorknoten auf die eingehenden Daten eine Aggregationsfunktion an. Das aus dieser resultierende Aggregat wird an Stelle der einzelnen Eingangsdaten weitergeleitet.

Das Beispiel in Abbildung 1.1 zeigt ein typisches Szenario, in dem fünf Sensorknoten die Raumtemperatur zweier Räume messen. Die Basisstation kann mit den messenden Sensorknoten A bis E, in der Abbildung weiß, aufgrund begrenzter Reichweite nicht direkt kommunizieren. Die Knoten X, Y und Z, in der Abbildung grau, müssen daher die Daten weiterleiten.

Ist die Basisstation dabei nur an dem Mittelwert der Raumtemperaturen interessiert, können die Knoten X, Y und Z eine Datenaggregation vornehmen. Die Aggregationsfunktion ist in diesem Fall eine Funktion, die aus allen Eingangsdaten den Mittelwert bildet. Die einzelnen Messwerte der Knoten A bis E werden so nicht unverändert weitergeleitet, sondern durch die Knoten X bis Z aggregiert. Knoten X leitet ausschließlich den Mittelwert der Daten von Knoten A und B weiter. Knoten Y leitet den Mittelwert der Knoten C bis E weiter. Schließlich bildet Z aus den bereits aggregierten Daten von X und Y nochmals den Mittelwert und leitet diesen an die Basisstation weiter. Das verkleinert die zu übertragende Datenmenge und spart Energie.

Nicht nur die begrenzten Ressourcen, auch die Sicherheit stellt die Forscher vor neue Herausforderungen. Sensornetze kommunizieren drahtlos und sind deswegen potentiellen Lauschangriffen und – hier unterscheiden sie sich von den uns vertrauten Funk-Netzwerken, wie beispielsweise „Wireless Lan“ – möglicherweise dem physikalischen Zugriff eines Angreifers ausgesetzt. Dieser könnte Sensorknoten manipulieren und so falsche Daten in das Netzwerk einbringen [5]. Der Nutzer eines Sensornetzes dagegen möchte authentische Messwerte erhalten, von denen er sich sicher ist, dass sie nicht manipuliert wurden.

Auf das Beispiel in Abbildung 1.1 angewendet, bedeutet dies, dass sich die Basisstation über den Ursprung und die Korrektheit der empfangenen Daten sicher sein muss. Durch aggregierende Knoten, die zwischen der Basisstation und den messenden Knoten liegen, kann jedoch Einfluss auf die Daten genommen werden. Der Basisstation ist es bei Verwendung von Aggregation nicht möglich zu überprüfen, ob die aggregierenden Knoten fehlerfrei arbeiten. Zum Beispiel kann Knoten Y, von einem Angreifer manipuliert, einen völlig falschen Mittelwert für den durch die Knoten C bis E überwachten Raum melden. Z und die Basisstation können nicht feststellen, ob das gemeldete Aggregat korrekt ist. Dazu wäre ein direkter Zugriff auf die Knoten C bis E notwendig.

Die Forderung nach authentischen Daten auf der einen Seite und Energieeffizienz durch Aggregation auf der anderen Seite führt damit zu dem Begriff der authentischen Aggregation. Eine authentische Aggregation vereint beide Konzepte und ermöglicht Energieeinsparungen ohne die Authentizität der Daten zu vernachlässigen.

1.1 Zielsetzung der Arbeit

Aufgrund der gegensätzlichen Anforderungen die Aggregation und Authentizität stellen, ist die Realisierung einer authentischen Aggregation nicht einfach. Ziel dieser Arbeit ist daher die Realisierung einer authentischer Aggregation in einem drahtlosen

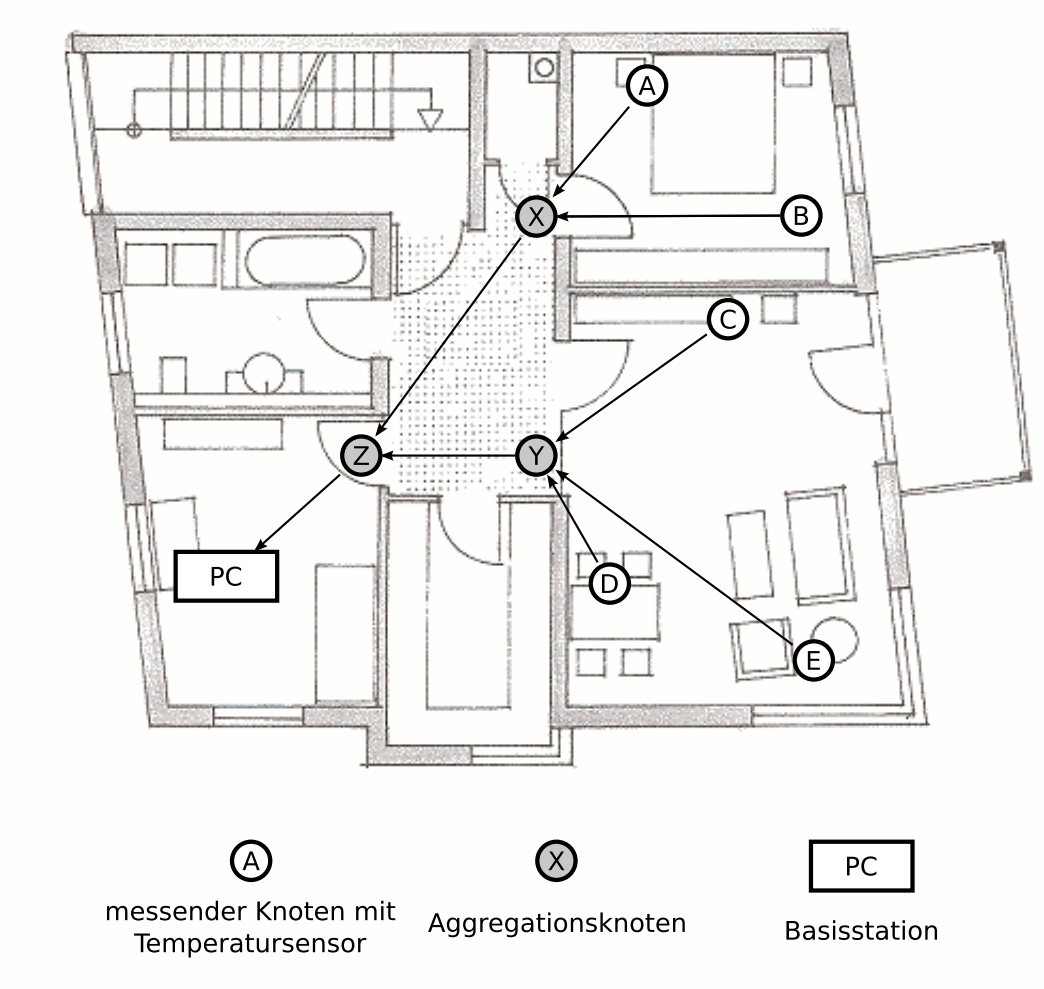


Abbildung 1.1: Anwendungsszenario für Datenaggregation

Sensornetz. Dabei baut diese Arbeit auf Ergebnissen der Studienarbeit „Authentischer und effizienter Datentransport in drahtlosen Sensornetzen“ [41] und einem zugehörigen Konferenzartikel [7] auf. Das in der Studienarbeit entwickelte Protokoll ESAWN wird kurz beschrieben und seine Nachteile erläutert. Beispielsweise kann ein Angreifer die Aggregation zum Abbruch bringen und er kann außerdem nicht gezielt identifiziert werden. Um dem zu begegnen schlägt diese Arbeit zwei neue Protokolle vor, ESAWN-2 und ESAWN-NR. ESAWN-2 ist weniger anfällig gegen Störungen eines Angreifers. ESAWN-NR ermöglicht die gezielte Identifikation der korrumpierten Knoten im Netzwerk. Dabei finden die gleichen Randbedingungen Beachtung, die bereits bei der Entwicklung von ESAWN festgesetzt wurden. Diese werden in Abschnitt 2.5 genauer definiert und begründet.

1.2 Gliederung der Arbeit

Diese Arbeit ist wie folgt gegliedert: In Kapitel 2 wird die Problemstellung der authentischen Aggregation detailliert geschildert und die Herausforderungen bei ihrer Realisierung aufgezeigt. Ein Blick in die Literatur gibt einen Überblick über aktuelle Forschungsarbeiten und deren Lösungsansätze. Es werden Randbedingungen präzisiert und Entwurfsziele formuliert, mit denen eine zufriedenstellende Lösung

zurecht kommen muss. Abgeschlossen wird mit einer Betrachtung und Bewertung des Protokolls ESAWN aus [41].

In Kapitel 3 wird auf diesen Erkenntnissen aufgebaut, um die potentielle Nachteile von ESAWN zu beheben. Dazu werden zwei neue Protokolle ESAWN-2 und ESAWN-NR vorgestellt und ihre Funktionsfähigkeit bewiesen.

Alle drei Protokolle wurden in einer Simulationsumgebung und einer realen Sensorknotenplattform implementiert und evaluiert. Eine Beschreibung der Vorgehensweise und die Ergebnisse der Evaluation finden sich in den Kapiteln 4 und 5. Die Evaluation wird dabei die Leistungsfähigkeit und Effizienz der Protokolle in verschiedenen Situationen aufzeigen sowie Vor- und Nachteile benennen.

Der Anhang am Ende dieser Arbeit legitimiert einige Vereinfachungen, die bei den Simulationen und deren Betrachtung vorgenommen wurden. Außerdem bietet er eine Übersicht, über die verwendete Notation.

2. Analyse

Ziel des Kapitels: Dieses Kapitel liefert die Motivation für die Beschäftigung mit der authentischen Aggregation. Dazu wird aufgezeigt, dass Authentizität und Aggregation orthogonale Anforderungen an ein Sensornetz darstellen, die nicht problemlos miteinander verbunden werden können. Ein Blick auf ESAWN und andere aktuelle Forschungsarbeiten skizziert die dort vorgeschlagenen Lösungen, stellt Vor- und Nachteile heraus und bildet damit die Motivation für einen neuen Lösungsansatz.

Aufbau des Kapitels: Im Abschnitt 2.1 wird eine einheitliche Notation für diese Arbeit festgelegt, um darauf folgend in Abschnitt 2.2 wichtige Begrifflichkeiten formal zu definieren. Es folgen in Abschnitt 2.3 zuerst einige Annahmen, die getroffen wurden, insbesondere zum Aufbau des Sensornetz und der Stärke des Angreifers. Nach der Beschreibung der Problematik einer authentischen Aggregation in Abschnitt 2.4, werden in Abschnitt 2.5 die Entwurfsziele einer Lösung beschrieben. Mit einem Überblick über Forschungsarbeiten in Abschnitt 2.6 schließt das Kapitel.

2.1 Zur Notation

In dieser Arbeit werden durchweg Symbole und Abkürzungen für die unterschiedlichen Dimensionen eines Sensornetz und die Parameter der Kommunikationsprotokolle verwendet. Zur Referenz bietet die Tabelle B.1 in Anhang einen Überblick und gibt Verweise auf die Abschnitte, in denen das Symbol erstmals eingeführt wird.

2.2 Definitionen

Dieser Abschnitt bietet eine formale Definition verschiedener, in dieser Arbeit verwendeter, Begrifflichkeiten. Um einen schnellen Überblick zu gewährleisten, sind die definierten Begriffe **fett** hervorgehoben.

Sensornetz

Ein **Sensornetz** ist ein drahtloses Netzwerk bestehend aus Sensorknoten, die ihre Umgebung sensorisch überwachen und sich spontan miteinander vernetzen können

um diese Informationen auszutauschen und weiterzuleiten. Es gibt keine festgelegte Infrastruktur in einem Sensornetz, jeder Sensorknoten ist in Ausstattung und Fähigkeiten gleich.

Ein Sensorknoten ist **korrumpiert**, wenn es einem Angreifer gelungen ist, physisch Zugriff auf den Knoten zu nehmen und ihn umzuprogrammieren um Einfluss auf das Sensornetz zu nehmen. Ansonsten wird der Sensorknoten als **legitim** bezeichnet. Ein korrumpierter Knoten wird versuchen, ein gefälschtes Aggregat erfolgreich in das Netzwerk einzubringen, ohne dass dies entdeckt wird. Gelingt es ihm, war sein **Betrugsversuch** erfolgreich.

Datenaggregation

Datenaggregation, oder kurz Aggregation, bezeichnet das Zusammenfassen von Daten mittels einer Aggregationsfunktion. Ziel ist die kompaktere, jedoch verlustbehaftete Darstellung der Daten.

- Eine **(Daten-)Aggregation** besteht aus
  - einem **Aggregationsknoten** A, der die Aggregation durchführt,
  - δ Sensorknoten (Quellsensoren) N_1, N_2, …, N_δ die ihre Daten d_1, d_2, …, d_δ (Quelldaten) an den Aggregationsknoten A liefern,
  - eine **Aggregationsfunktion** f, die beschreibt, wie aus den Quelldaten d_1, d_2, …, d_δ das Aggregat agg_A berechnet wird,
  - das **Aggregat** $agg_A = f(d_1, d_2, \ldots, d_\delta)$, das Ergebnis der durchgeführten Aggregation.

- Aggregation kann **kaskadieren**. Das ist der Fall, wenn ein Datum d_i, welches Eingangswert für eine Aggregation ist, bereits selbst ein Aggregat darstellt.

- Ein Aggregat agg_A ist **authentisch**, wenn $agg_A = f(d_1, d_2, \ldots, d_\delta)$ gilt und die Quelldaten d_1, d_2, …, d_δ von den zugehörigen Quellsensoren N_1, N_2, …, N_δ stammen.

- Eine **authentische Aggregation** liegt dann vor, wenn ein Sensorknoten im Aggregationsbaum sein authentisches Aggregat kennt.

- Die **Wahrscheinlichkeit korrekter Aggregation (WKA)** bezeichnet die Wahrscheinlichkeit, dass in einem bestimmten Sensornetz mit korrumpierten Knoten *alle* Aggregationen authentisch verlaufen.

Paketaggregation

Im Unterschied zur Datenaggregation handelt es sich bei der **Paketaggregation** um ein verlustfreies Bündeln von Daten. Jedes Paket besteht aus einem **Paketkopf** mit Steuerdaten und der **Nutzlast** mit den Anwendungsdaten. Statt jedes Datum einzeln zu versenden, werden mehrere Daten zusammengefasst und so die Nutzlast eines Datenpakets besser ausgenutzt. Dadurch müssen bei gleicher Datenmenge weniger Paketköpfe übertragen werden.

Aggregationsbaum

- Verarbeitet ein Aggregationsknoten A Daten eines anderen Knotens N_i, so besteht zwischen beiden Knoten eine **Aggregationsbeziehung**. Alle Aggregationsbeziehungen können in einem Graphen, dem so genannten Aggregationsbaum, dargestellt werden.

- Ein **Aggregationsbaum** ist ein gerichteter, zyklenfreier, zusammenhängender Graph $G = (V, E)$, in dem jeder Sensorknoten durch eine Ecke $e \in E$ und jede Aggregationsbeziehung durch eine gerichtete Kante $v \in V$ in Richtung des Aggregationsknotens dargestellt wird.

- Die Zahl der eingehenden Kanten eines Aggregationsknoten wird als **Verzweigungsgrad** δ bezeichnet. Der **mittlere Verzweigungsgrad** eines Aggregationsbaums gibt an, wieviele Daten ein Aggregationsknoten im Durchschnitt zu einem Aggregat zusammenfassen muss.

- **Blattknoten**, alle Knoten mit $\delta = 0$, übernehmen dabei die Aufgabe der sensorischen Umwelterfassung. Ihr Messergebnis teilen sie ihrem Vaterknoten im Aggregationsbaum mit.

- **Aggregationsknoten**, alle Knoten mit $\delta > 0$, empfangen solche Messdaten, aggregieren sie und leiten sie an ihren Vaterknoten weiter.

- Jedem Knoten kann ein **Aggregationspfad** zugewiesen werden, der Weg, den seine Messdaten bis zur Wurzel nehmen.

- Existiert von Knoten v zu Knoten u eine gerichtete Kante, so ist u der **1-Vorgänger** (oder **direkte Vorgänger**) von v.

- Der r**-Vorgänger** eines Knotens v ist der Vorgänger des $(r-1)$-Vorgängers. Er besitzt einen **Abstand** von r zu Knoten v.

- Einen besonderen Aggregationsknoten stellt die **Senke**, die Wurzel des Aggregationsbaums, dar. Sie bildet aus den empfangenen Daten das Gesamtaggregat des Sensornetz und teilt es der **Basisstation**, einem PC oder einer ähnlichen Steuereinheit, mit.

- Der längste Pfad von einem Blattknoten zur Senke bestimmt die **Höhe** h des Aggregationsbaums.

Vollständigkeit und Korrektheit

Die ESAWN-Protokolle werden im späteren Verlauf auf ihre Vollständigkeit und Korrektheit hin untersucht. Beide Begriffe sind in Anlehnung ihrer Bedeutung in der mathematischen Logik gewählt. Ein Protokoll zur authentischen Aggregation ist dabei **vollständig**, wenn es jeden durch korrumpierte Knoten versuchten Betrug entdeckt. Es ist **korrekt**, wenn es nur dann einen Betrug feststellt, wenn es diesen auch tatsächlich gegeben hat.

Non-Repudiation

Unter **„Non-Repudiation“**, auf deutsch Nicht-Abstreitbarkeit, wird eine Protokolleigenschaft bezeichnet, die es ermöglicht, zu überprüfen, ob ein Datenpaket wirklich von einem bestimmten System versandt („Non-Repudiation of origin“) oder empfangen („Non-Repudiation of delivery“) wurde [22]. Im Rahmen dieser Arbeit ist besonders die erste Form interessant, die es ermöglichen wird, einem Sensorknoten nachzuweisen, dass er ein bestimmtes Datum tatsächlich versandt hat.

Begriffe der ESAWN-Protokolle

- Ein Sensorknoten ist n-ter **Zeuge** für ein Aggregat agg_A, wenn er die von A durchgeführte Aggregation zu Kontrollzwecken selbst durchführt um das Ergebnis, agg_A^n, entweder selbst auszuwerten oder einem anderen Knoten zur Verfügung zu stellen.
- In einer Liste von n Werten, kommt ein Wert **mehrheitlich** vor, wenn er *mehr* als $\frac{n}{2}$-mal vertreten ist.
- Ein Sensorknoten ist **Entscheider** für ein Aggregat agg_A, wenn er aus einer Liste von n berechneten Aggregaten agg_A^0, agg_A^1, …, agg_A^n den mehrheitlich vorkommenden Wert bestimmt.
- Ein Sensorknoten kann gleichzeitig die Aufgabe eines Zeugen, eines Entscheiders und eines Aggregationsknotens für verschiedene Aggregate übernehmen.
- Ein Sensorknoten ist für ein Aggregat agg_A **zuständig**, wenn es sich um den Knoten A selbst oder einen seiner zugewiesenen Zeugen handelt.

2.3 Randbedingungen und Voraussetzungen

Dieser Abschnitt beschreibt die getroffenen Annahmen und unter denen der Lösungsvorschlag zur sicheren Aggregation funktionieren muss. Diese Annahmen beziehen sich auf den Aufbau des Netzwerkes und den erwarteten Angreifer.

2.3.1 Aufbau des Netzwerks

- Es wird von einem Netzwerk mit n Knoten ausgegangen. In diesem Netzwerk wird ein Aggregationsbaum mit einem mittleren Verzweigungsgrad δ aufgebaut. Ein beliebiger Knoten in solch einem Aggregationsbaum hat damit im Mittel δ Kindknoten. Der gesamte Baum hat eine Höhe von

$$h = \log_\delta(1 + n \cdot (\delta - 1)) - 1 \qquad [6].$$

- Das Protokoll zum Aufbau des Aggregationsbaums wird als gegeben betrachtet, siehe beispielsweise [20; 43]. Jedem Sensorknoten ist seine Lage im Aggregationsbaum bekannt. Für eine konventionelle Aggregation muss der Knoten seine direkten Kindknoten und seinen Vaterknoten kennen. Für eine sichere Aggregation mit ESAWN braucht der Knoten noch Informationen über weitere Knoten. Dies wird noch im Detail angesprochen.

- Jeder Knoten hält für jeden anderen Knoten, mit dem er kommunizieren muss, jeweils einen symmetrischen Schlüssel bereit. Die Verteilung der Schlüssel erfolgt durch einen geeigneten Mechanismus, zum Beispiel SKEY aus [6] oder auch [14; 8].

  Damit Knoten gezielt miteinander in Verbindung treten können, wird jedem Sensorknoten eine ID zugewiesen, über die er adressierbar ist. Diese ID wird zur Identifizierung und Kommunikation genutzt und ermöglicht außerdem die Wahl des korrekten Kommunikationsschlüssels und die Verwaltung und Zuordnung der empfangenen Daten. Der Senke ist dabei immer die ID 0 zugewiesen.

- Es wird vorausgesetzt, dass jeder Knoten mit seinen Kindknoten und dem Vaterknoten im Aggregationsbaum direkt kommunizieren kann. Weitere direkte Kommunikationsverbindungen werden nicht vorausgesetzt. Pakete im Sensornetz werden damit entlang der Kanten des aufgebauten Aggregationsbaums „geroutet“ ohne dass dabei Knoten übersprungen werden. Die Übertragung der Pakete im Netzwerk soll zuverlässig (beispielsweise durch „Stop and Wait“) erfolgen, so können durch verlorene Pakete auftretende Probleme ausgeschlossen werden.

2.3.2 Angreifermodelle

Sensornetze bieten aufgrund ihrer Struktur vielfältige Angriffsmöglichkeiten. Sowohl der oft freie physische Zugang zu einzelnen Knoten, als auch die drahtlose Kommunikation stellen Sicherheitsrisiken dar. Sensorknoten sind in der Regel nicht vor physischen Manipulationen geschützt. In der Literatur werden zur Analyse der Sicherheitsrisiken unterschiedliche Angreifermodelle angenommen [25]. Dabei wird unter anderem eine Einteilung in passive und aktive Angreifer vorgenommen.

Passive und aktive Angreifer

Die Übertragung von Daten in Sensornetzen erfolgt per Funk. Passive Angreifer sind in der Lage, diesen Netzwerkverkehr zwischen den einzelnen Sensorknoten abzuhören. Obwohl die Übertragungsreichweite der Knoten begrenzt ist, geht man in der Regel davon aus, dass der Angreifer vom gesamten Netzwerkverkehr Kenntnis erlangen kann und betrachtet damit den schlimmsten anzunehmenden Fall.

Aktive Angreifer erlangen im Gegensatz zu passiven Angreifern auch physischen Zugriff auf einen oder mehrere Knoten des Sensornetz. Dadurch können sie den gesamten Speicher dieser Knoten auslesen und erhalten so Kenntnis über darin gespeicherte Informationen wie Messdaten und kryptographische Schlüssel. Dies ermöglicht es ihnen, gefälschte Pakete in das Netzwerk einzubringen.

Bei Sensornetzen ist es, im Gegensatz zu einem konventionellen Funk-Netzwerk, aufgrund der physischen Zugriffsmöglichkeit unvermeidlich, auch aktive Angreifer bei der Sicherheitsanalyse zu berücksichtigen.

Statische und dynamische Angreifer

Eine weitere Unterscheidungsmöglichkeit liegt in der Vorgehensweise des Angreifers. Ein statischer Angreifer korrumpiert zu einem festen Zeitpunkt eine Menge von Knoten. Ein dynamischer Angreifer hingegen kann die Menge der korrumpierten Knoten auch im nachhinein ändern oder erweitern und ist damit flexibler.

Globaler und lokaler Angreifer

Eine dritte Unterscheidungsmöglichkeit bezieht sich auf den Aktionsradius des Angreifers. Während ein lokaler Angreifer nur ein räumlich begrenzten Teil des Sensornetzes korrumpieren kann, ist der globale Angreifer nicht derart eingeschränkt und kann eine beliebige Teilmenge der Sensorknoten, unabhängig von ihrer Lage, korrumpieren. Ein globaler Angreifer ist damit als mächtiger einzustufen als der lokale Angreifer.

2.3.2.1 Angreifermodell für drahtlose Sensornetze

Für alle folgenden Betrachtungen wird daher von einem sehr starken, für drahtlose Sensornetze durchaus realistischen, Angreifermodell ausgegangen: einem statischen globalen aktiven Angreifer.

Dabei wird angenommen, dass der Angreifer aus der Menge $\mathcal{N}$ der Knoten eines Sensornetz eine beliebige aber feste Untermenge $\mathcal{B} \subset \mathcal{N}$ korrumpiert (aktiv und statisch). Die Auswahl der korrumpierten Knoten soll dabei zufällig gleichverteilt und nicht lokal beschränkt sein (global). Die Größe dieser Menge $|\mathcal{B}|$ ist durch die Kosten beschränkt, die der Angreifer für die Korrumpierung des Netzwerks aufbringen will oder kann und damit stark vom Anwendungsszenario abhängig [5]. $\mathcal{B}$ ist während des Protokollablaufs statisch und ändert sich nicht [11]. Der Anteil korrumpierter Knoten im Netzwerk berechnet sich mit $\beta = \frac{|\mathcal{B}|}{n}$.

Ebenso wird ausgeschlossen, dass ein Angreifer das Netz durch gezieltes Deaktivieren einzelner Sensorknoten stört. Solche klassischen „Denial-of-Service“-Angriffe werden im Rahmen dieser Arbeit nicht betrachtet. Ziel des Angreifers ist das Abhören von Netzwerkdaten sowie das Einbringen von Fehlinformationen in das Netzwerk.

Im Sensornetz wird, wie in Abschnitt 2.3.1 beschrieben, ein Aggregationsbaum aufgebaut. Es wird davon ausgegangen, dass die Blattknoten dieses Aggregationsbaums korrekte Messwerte liefern. Diese Annahme ist zwingend notwendig, da die Korrektheit eines einzelnen Messwerts nicht überprüft werden kann. Der Knoten könnte bewusst falsche Aussagen über seinen Messwert machen, aber auch durch äußere Einflüsse (beispielsweise das gezielte Erwärmen der Umgebung bei einem Temperatursensor) beeinflusst werden.

Weiterhin wird die Senke des Netzwerks, der Wurzelknoten des Aggregationsbaums, als legitim angesehen, da dies der einzige Knoten im Netzwerk ist, auf den der Nutzer direkten Zugriff hat und über den Befehle in das Netzwerk geleitet und Aggregate empfangen werden.

2.4 Authentische Aggregation

Die Sicherheitsgarantien eines authentischen Datentransports und die Effizienz eines aggregierenden Datentransports lassen sich nur schwer vereinen. Will die Senke die Authentizität des Aggregats überprüfen, braucht sie direkten Zugriff auf die Quelldaten der Aggregation. Die Blattknoten müssen also direkt mit der Senke kommunizieren.

Durch einen oder mehrere Aggregationsschritte auf dem Weg zur Senke ist diese Möglichkeit jedoch nicht mehr gegeben. Die Senke kann die Authentizität der Daten

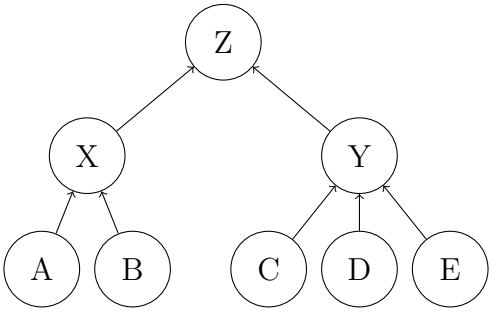


Abbildung 2.1: Formale Darstellung des Beispiels aus Abbildung 1.1

nur sicherstellen, wenn sie den zwischenliegenden Aggregationsknoten vertraut. Diese aggregieren die Daten und verändern sie damit so, dass sie keinen Schluss mehr auf die Originaldaten zulassen. In Sensornetzen kann, aufgrund des Angreifermodells aus Abschnitt 2.3.2.1, nicht von solch einem vorhandenen Vertrauen ausgegangen werden, daher ist so keine Authentifikation mehr möglich.

Kommunizieren dagegen Blattknoten und Senke direkt miteinander, können die Aggregationsknoten dazwischen nicht auf die verschlüsselten Daten zugreifen und so auch keine Aggregation durchführen. Die Senke stellt so zwar die Authentizität der Daten sicher, verzichtet jedoch auf die Vorteile der Datenaggregation, insbesondere die damit verbundene Energieeinsparung.

Ein Blick auf das Beispiel aus der Einführung (Abbildung 1.1) veranschaulicht das Problem. Der dort dargestellte Aggregationsbaum ist in Abbildung 2.1 formaler dargestellt.

Entweder verschlüsseln die Messknoten A bis E ihre Messdaten direkt für die Senke. Die Knoten X, Y und Z können diese Daten lediglich weiterleiten, aber nicht aggregieren, da sie keinen Zugriff auf die Klartext-Daten haben. Dafür kann die Senke den Ursprung der Daten verifizieren, indem sie beispielsweise die Messknoten zusätzlich zu dem Messwert eine Prüfsumme o.ä. mitverschlüsseln lässt, die eine Unversehrtheit des Pakets sicherstellt.

Die andere Möglichkeit besteht darin, den Aggregationsknoten Zugriff auf die Klartextdaten zu gewähren. Die Messknoten verschlüsseln dazu ihre Daten nicht mehr für die Senke sondern für den Vaterknoten im Aggregationsbaum. Knoten A verschlüsselt beispielsweise für Knoten X. Dieser empfängt von A und B die Messwerte, kann die Pakete entschlüsseln und die Daten aggregieren. Danach verschlüsselt X die Daten für Z und sendet sie an diesen Knoten weiter. Das Gleiche geschieht für die Knoten C bis E und den Aggregationsknoten Y. Dadurch, dass nun weitere Knoten Zugriff auf die Messwerte und Aggregate erhalten haben, kann die Senke die Authentizität des Aggregats (das heißt, dass es wirklich das Aggregat der Messwerte von den Messknoten ist) nicht mehr sicherstellen. Knoten Z könnte gegenüber der Senke beliebige Werte als Aggregat melden. Die Senke kann die Korrektheit dieser Aussage nicht überprüfen ohne mit den Blattknoten Kontakt aufzunehmen.

Im Rahmen der Studienarbeit [41] und der Arbeit mit ESAWN wurde klar, dass eine authentische Aggregation unter diesen Bedingungen ein Kompromiss zwischen den zwei gegensätzlichen Anforderungen des aggregierenden und des authentischen Datentransport sein muss. Authentische Aggregation wird daher Authentizität bieten

und Aggregation nutzen, wenn auch in einer abgestuften Form. Für Authentizität bedeutet dies ein Loslösen von der Vorstellung 100%-iger Sicherheit zugunsten der Nutzung von Aggregation. Für Aggregation bedeutet es Verzicht auf einen Teil der Einsparung, die durch Aggregation erzielt werden kann, zugunsten eines „gewissen Grades“ an Authentizität. Was genau unter dieser relaxierten Form der Authentizität zu verstehen ist, wird später genauer beschrieben.

2.5 Anforderungen an eine Lösung

Das Ziel besteht daher darin, ein Protokoll zu entwickeln, das beide Anforderungen berücksichtigt und damit den Ansprüchen eines authentischen Datentransports gerecht wird ohne auf die Effizienz eines aggregierenden Datentransports zu verzichten.

Im Einzelnen ergeben sich an ein Protokoll zur authentischen Aggregation, sowie das zugrundeliegende Sensornetz und dessen Sensorknoten folgende Anforderungen.

- Vertraulichkeit und Authentizität des Datenverkehrs.
  Das Sensornetz muss gegen einen statischen, globalen und aktiven Angreifer gesichert sein. Dem Angreifer soll es weder ermöglicht werden, Datenverkehr zwischen den Knoten abzuhören, noch soll es ihm möglich sein, gefälschte Pakete in das Netzwerk einzuspeisen und so dessen Betriebsablauf zu stören oder Messergebnisse zu verfälschen.

- Ausschließliche Nutzung symmetrischer Kryptographie.
  Die Verwendung von asymmetrischer Kryptographie ist komplex und in in Sensornetzen zeit- und energieaufwendig [16; 9]. Der Einsatz von effizienter symmetrischer Kryptographie muss zur Realisierung des Protokolls ausreichen.

- Nutzung von Datenaggregation.
  Die Funkübertragung von Daten ist eine der energieaufwendigsten Aktionen jedes Sensorknotens [29]. Mittels Datenaggregation muss das zu übertragene Datenvolumen reduziert werden. Durch die damit erzielte Energieeinsparung wird eine längere Lebenszeit des Sensornetz ermöglicht.

- Nutzung beliebiger Aggregationsfunktionen.
  Aggregation ist in einem Sensornetz oft nicht nur auf das Bilden von Summen und Durchschnitten beschränkt. Eine optimale Lösung ist daher unabhängig von der oder den verwendeten Aggregationsfunktionen einsetzbar.

- Parametrisierung des Authentizität-Energie-Tradeoffs.
  In Abschnitt 2.4 wurde aufgezeigt, dass eine Authentische Aggregation nur durch Abstriche auf beiden Seiten, der Authentizität der Daten und der Energieeinsparungen durch Aggregation, realisiert werden kann. Dieser Tradeoff sollte szenariobezogen einstellbar sein um jeweils den gewünschten Grad an Authentizität und Energieverbrauch zu erreichen.

- Erkennen von Betrugsversuchen.
  Durch den Angreifer korrumpierte Aggregationsknoten können versuchen gefälschte Daten in das Sensornetz einzuspeisen. Jeder legitime Knoten der ein gefälschtes Aggregat erhält, muss dies erkennen können und entsprechend behandeln. ESAWN beispielsweise stoppt den Aggregationsvorgang und verschickt eine „Alarm“-Meldung an alle Knoten.

- Identifikation der korrumpierten Knoten.
  Erhält ein Knoten B von Knoten A eine Nachricht, so kann er die Authentizität dieser Nachricht selbst überprüfen und kann sich sicher sein, dass die Daten tatsächlich von B stammen. Stellt B einen Betrugsversuch von A fest, kann er dies nicht ohne weiteres einem dritten Knoten mitteilen. Der dritte Knoten kann nicht überprüfen ob die Meldung von A korrekt ist. Dies wäre aber für einen Ausschluss des korrumpierten Knoten A aus dem Netzwerk notwendig. Durch „Non-Repudiation“ kann sichergestellt werden, dass A nachgewiesen werden kann, was genau er versendet hat.

- Einheitliche Sensorknoten.
  Es darf keine Notwendigkeit speziell ausgezeichneter Sensorknoten, zum Beispiel eine besonders leistungsstarke Senke oder ähnliches, geben, die beim Aufbau des Netzwerks berücksichtigt werden muss. Jeder Sensorknoten ist mit der gleichen Leistungsfähigkeit und Speicherkapazität ausgestattet. Dies erleichtert das Auslegen des Sensornetz, da beliebige Aggregationsbäume aufgebaut werden können, ohne auf besondere Fähigkeiten einzelner Sensorknoten achten zu müssen.

2.6 Stand der Forschung

Nach einem Überblick über andere Forschungsarbeiten zu dem Thema des authentischen und aggregierenden Datentransports wird im zweiten Teil dieses Abschnitts die Protokollidee von ESAWN skizziert und die Nachteile von ESAWN benannt.

2.6.1 Literaturüberblick

In letzter Zeit haben sich immer mehr Forschungsarbeiten mit den Themen sichere Kommunikation, Aggregation und insbesondere deren Verbindung miteinander beschäftigt.

In der Arbeit [44] wird ein Protokoll vorgeschlagen, welches bei bis zu b korrumpierten Knoten, jedes von solch einem Knoten gefälschte Datenpaket aufdeckt. Dazu wird gefordert, dass jeder gemessene Sensorwert von mindestens $b + 1$ Knoten bestätigt wird. So wird eine gefälschtes Datum erfolgreich identifiziert und gefiltert, der dabei entstehende Zusatzaufwand ist jedoch hoch und erfordert untere anderem mehr Kommunikation und damit Energie. Der Einsatz von Aggregation wird nicht betrachtet. Die Idee, Aggregate redundant zu berechnen, wird bei ESAWN aufgegriffen.

In [13] wird symmetrische Verschlüsselung eingesetzt, um das Netzwerk gegen einen passiven Angreifer zu schützen. Jeder Aggregationsknoten entschlüsseln empfangene Pakete, berechnet das Aggregat der entschlüsselten Daten und sendet das Ergebnis, mit einem neuen Schlüssel codiert, an seinen Vaterknoten im Aggregationsbaum weiter. Ein einziger korrumpierter Knoten könnte so allerdings erfolgreich betrügen. Die Arbeit weist auf dieses Problem hin und schlägt vor, einen externen Mechanismus zu nutzen, der solche Knoten erkennt und ausschließt, ohne diesen genauer zu beschreiben.

In der Arbeit [27] beschäftigt sich mit diesem Problem detaillierter. Ein Aggregationsknoten wird hier durch ein Verfahren überprüft, das abhängig von der zum Einsatz kommenden Aggregationsfunktion ist. Bei der Aggregationsfunktion Mittelwertberechnung werden beispielsweise Stichproben aus dem Datenbestand, der aggregiert wurde, zur Verifikation zu Rate gezogen. Dadurch kann sichergestellt werden, dass das Ergebnis mit einer bestimmten Wahrscheinlichkeit innerhalb eines vorgegebenen Intervalls um den korrekten Aggregationswert liegt. Kleine Manipulationen werden so selten entdeckt, eine größere Abweichung wird jedoch mit hoher Wahrscheinlichkeit aufgedeckt. Ein Nachteil ist, dass das Prüfverfahren abhängig von der zum Einsatz kommenden Aggregationsfunktion (zum Beispiel Mittelwertsberechnung) ist und sich nicht universell erweitern lässt.

Ein gänzlich anderer Ansatz ist die Verwendung von „privacy homomorphisms“. Sie kommen in [39], [10] und [1] zum Einsatz. Unter solch einem Homomorphismus ist dabei eine Verschlüsselungsfunktion zu verstehen, mit der bestimmte Rechenoperationen direkt auf den verschlüsselten Daten durchgeführt werden können. Ein Knoten kann dann eine Aggregation auf Daten durchführen, die ihm nicht im Klartext vorliegen. Er erlangt so auch keine Kenntnis von dem Inhalt, auch nicht von dem selbst berechneten Aggregat. Eine große Einschränkung dieses Verfahrens liegt darin, dass es nur auf die kleine Gruppe linearer Aggregationsfunktionen anwendbar ist, beispielsweise der Addition oder Multiplikation.

SAWN [18], ausgeschrieben „Secure Aggregation for Wireless Networks“, ist ein Protokoll, das verzögerte Aggregation und Authentizität nutzt, um das Ziel einer authentischen Aggregation zu erreichen. Von „verzögerter Aggregation“ spricht man, wenn Daten nicht direkt von dem Vaterknoten aggregiert werden, sondern dieser die Daten unverändert weiterleitet. Die eigentliche Aggregation erfolgt erst später. Bei SAWN findet sie im darauf folgenden Knoten, dem Großvaterknoten, statt. Von „verzögerter Authentifikation“ spricht man, wenn die Authentizität einer Nachricht nicht direkt beim Empfangen der Nachricht überprüft wird oder werden kann, sondern erst nach einer bestimmten zeitlichen Verzögerung. Dies tritt hier auf, weil zur Überprüfung der Authentizität zusätzliche Informationen der Senke notwendig sind, die diese erst nach Abschluss eines Messvorgangs offenlegt. Da diese Informationen im Broadcast-Verfahren an die Sensorknoten verbreitet werden, wird angenommen, dass die Sendeleistung der Senke deutlich größer ist als die der einzelnen Sensorknoten, um alle Knoten direkt zu erreichen.

Vorteil von SAWN ist es, dass es mit beliebigen Aggregationsfunktionen zurecht kommt. Nachteile sind neben der Notwendigkeit einer leistungsstärkeren Senke auch die einer fehlenden Skalierbarkeit. Insbesondere bei großen Netzwerken kommt es zu sehr großen Broadcast-Nachrichten, die durch die Senke verschickt werden. Die Größe dieser Nachrichten hängt linear mit der Netzwerkgröße zusammen. Ein weiterer Nachteil liegt darin, dass zwei im Aggregationsbaum hintereinander liegende korrumpierte Knoten Einfluss auf die Aggregation nehmen können ohne dass dies entdeckt wird.

2.6.2 ESAWN

All den bisher vorgestellten Forschungsarbeiten ist gemeinsam, dass sie nur einen Teil der Anforderungen aus 2.5 erfüllen. Das Protokoll ESAWN kommt diesen Anforderungen näher. Es wurde erstmals im Rahmen einer Studienarbeit [41] veröffentlicht. Eine Zusammenfassung des Protokolls findet sich in [7]. Eine Beschreibung in

Pseudo-Code und der Beweis, dass ESAWN eine authentische Aggregation bietet, ist in [6] nachzulesen.

ESAWN basiert auf ein teilweises Aufheben der Aggregation durch zusätzliche Berechnung von Aggregationsergebnissen und dessen Vergleich durch Drittknoten. Diese Drittknoten werden Zeugen genannt. Sie führen, zusätzlich zum eigentlichen Aggregationsknoten, die gleiche Aggregation durch, um damit den zuständigen Knoten zu kontrollieren. Die gegenseitige Kontrolle von Sensorknoten ermöglicht es, Aggregation zu nutzen, obwohl es keine Aggregationsknoten gibt, von denen die Legitimität vorab bekannt ist.

Gelingt es einem Angreifer jedoch, alle Sensorknoten, die sich gegenseitig kontrollieren, zu korrumpieren, hat er das Kontrollsystem effektiv ausgeschaltet. ESAWN bietet deshalb die Möglichkeit, den Umfang dieser Kontrollmaßnahme zu parametrisieren. Durch den Parameter k wird festgelegt, wieviele korrumpierten Aggregationsknoten auf einem Aggregationspfad maximal hintereinander liegen dürfen, ohne dass ein Betrug möglich ist. Durch den Parameter p bietet ESAWN außerdem die Möglichkeit, die Wahrscheinlichkeit, dass ein Aggregationsknoten durch Zeugen geprüft wird, festzulegen. Durch eine Wahrscheinlichkeit von $p < 1$ wird zwar der Energieverbrauch reduziert, es steigt aber auch die Wahrscheinlichkeit, dass ein Betrug unentdeckt bleibt. Mit den Parameter k und p bietet ESAWN die Möglichkeit den Energie-Authentizitäts-Tradeoff szenariobezogen einzustellen. So kann, wenn benötigt, ein hoher Grad an Authentizität erzielt werden, der ein gewisses Maß an Energie kostet. In besonders ressourcenbeschränkten Umgebungen dagegen kann die Energieeinsparung erhöht werden, was eine Reduzierung der Wahrscheinlichkeit korrekter Aggregation mit sich bringt.

Entdeckt ein Zeuge einen fehlende Übereinstimmung beim Vergleichen der empfangenen Aggregate mit den selbst berechneten Werten, löst dieser einen Alarm aus und stoppt das Protokoll an dieser Stelle. Durch den Alarm werden alle Sensorknoten im Netzwerk über den Betrug informiert. Eine Fortführung der Aggregation ist nicht möglich, da nicht nachgewiesen werden kann, welches das korrekte Aggregat ist. Natürlich können auch korrumpierte Aggregationsknoten solch einen Alarm melden, obwohl kein Grund dafür vorliegt.

2.6.2.1 ESAWN am Beispiel

Ein ausführliches Beispiel mit einem gut 20 Knoten umfassenden, komplexen Aggregationsbaum findet sich in der Studienarbeit über ESAWN [41]. Um den Vergleich mit den im nächsten Kapitel folgenden Protokollvorschlägen zu erleichtern, wird in Abbildung 2.2 auf den Ablauf von ESAWN an einem stark vereinfacheten Aggregationsbaum mit $\delta = 1$ und $k = 1$ eingegangen.

Der Blattknoten N misst zuerst sensorisch einen Wert d_N, den er an seinen Vaterknoten und k Zeugen, in diesem Fall Knoten B, sendet. Beide berechnen hieraus agg_A. A sendet seine Version an B und C und erlaubt es so B den Wert von agg_A mit seiner Version agg'_A zu vergleichen. Bei fehlender Übereinstimmung löst der Knoten Alarm aus. Andernfalls wird die Aggregation fortgesetzt.

Zwei Ablaufdiagramme veranschaulichen den Nachrichtenaustausch. Im ersten Diagramm 2.3 wird der theoretische Nachrichtenaustausch dargestellt. Darunter sind die logischen Kommunkationsverbindungen zwischen Sender und Empfänger einer

Nachricht zu verstehen. Im zweiten Diagramm 2.4 wird der tatsächliche Nachrichtenaustausch dargestellt. Dieser unterscheidet sich von dem ersten Diagramm, da die Knoten zu versendende Daten bündeln um die Zahl der zu sendenden Datenpakete zu reduzieren (Paketaggregation).

2.7 Zusammenfassung

ESAWN erweist sich als Protokoll, das eine authentische Aggregation erlaubt, die den Anforderungen aus 2.5 weitgehend entspricht. Im Vergleich zu anderen Forschungsarbeiten ermöglicht ESAWN die Verwendung beliebiger Aggregationsfunktionen und ist durch die Systemparameter k und p flexibel an verschiedene Anwendungsszenarien anpassbar, die ein unterschiedlich hohes Maß an Authentizität und Energieverbrauch erfordern.

Trotzdem bleiben einige Nachteile. So kann schon ein einziger korrumpierter Knoten durch falsche Alarm-Meldungen einen Aggregationsvorgang des Netzwerks stören und so den Regelbetrieb im Sensornetz unterbinden. Gemäß der Definition in Abschnitt 2.2 ist ESAWN damit vollständig, jedoch nicht korrekt.

Im Falle eines entdeckten fehlerhaften Aggregats ist der Aggregationsvorgang außerdem an dieser Stelle gescheitert und kann nicht fortgesetzt werden, da nicht festgestellt werden kann, welche Aggregate korrekt und welche fehlerhaft sind. Damit lässt sich ein korrumpierter Knoten auch nicht zweifelsfrei identifizieren und aus dem Netzwerk ausschließen. Um einem Knoten den Versand eines falschen Aggregats nachzuweisen, ist „Non-Repudiation“ notwendig, die ESAWN nicht bietet.

Unklar ist außerdem, welche Anforderungen ESAWN an den Speicher eines Sensorknotens stellt. Im Rahmen der Studienarbeit über ESAWN blieb die Speicherkomplexität unbetrachtet, daher wird dies im Abschnitt 5.3.3 nachgeholt.

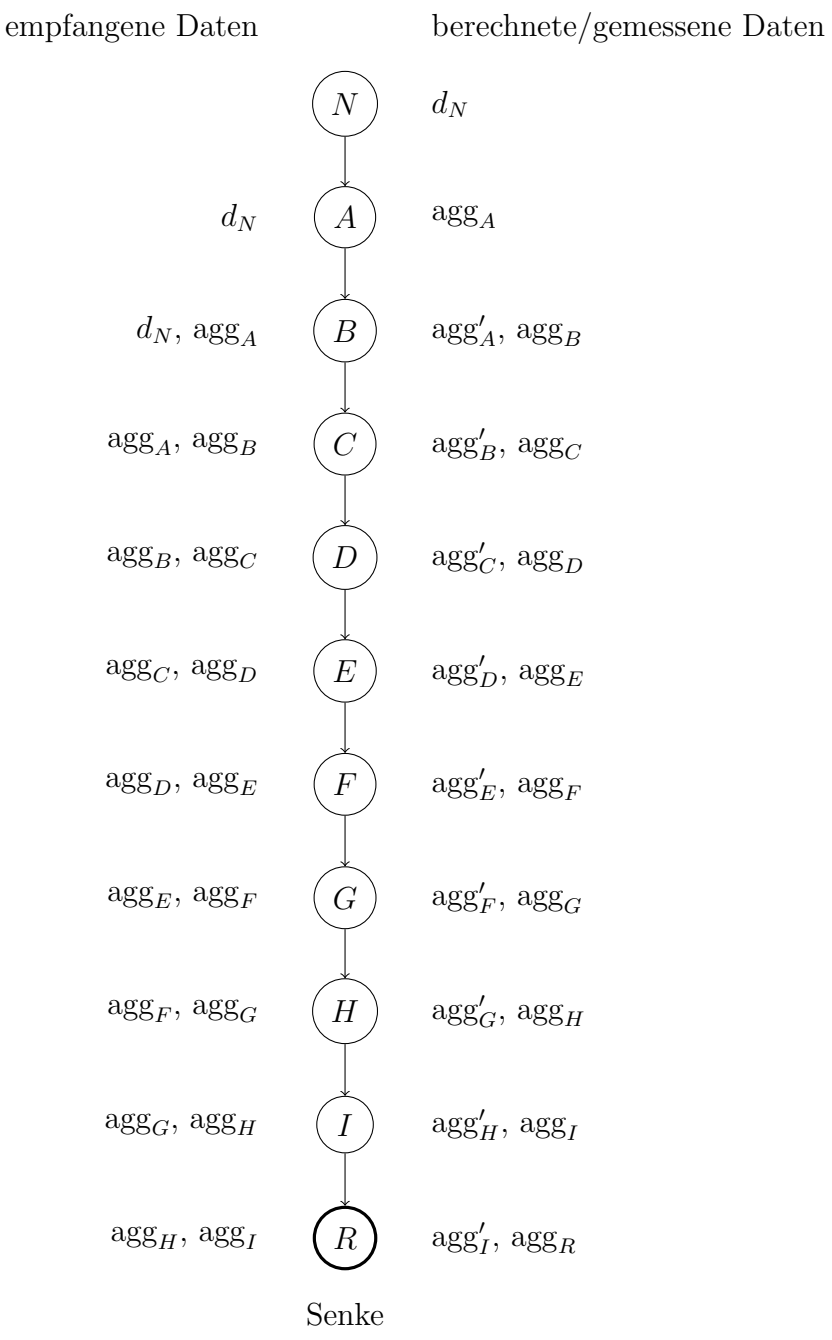


Abbildung 2.2: Protokollablauf bei ESAWN, $\delta = 1$ und $k = 1$

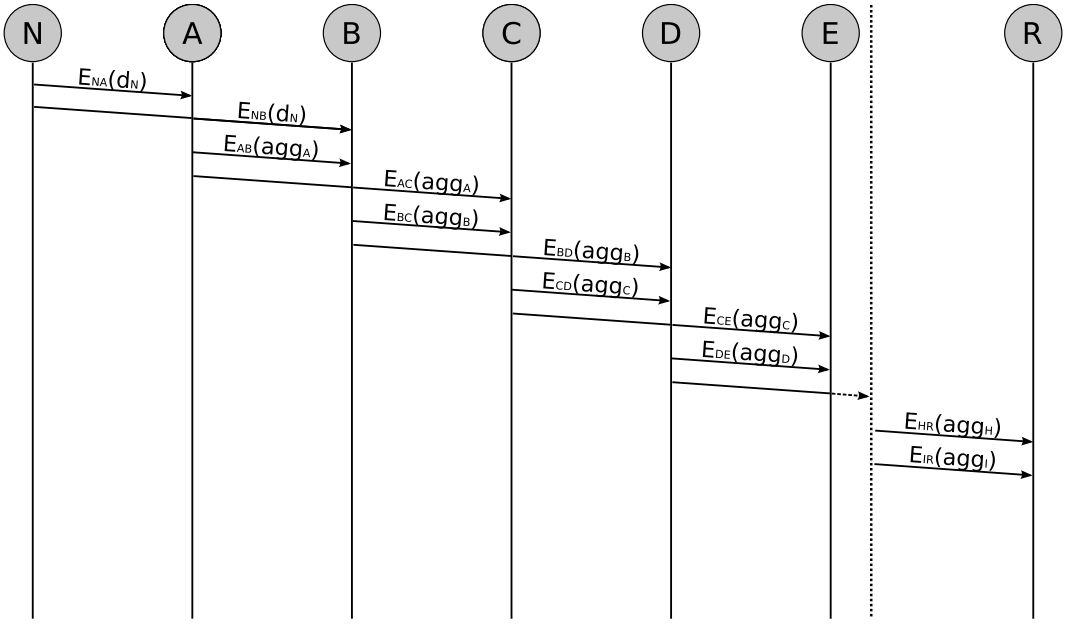


Abbildung 2.3: Visualisierung der theoretischen Kommunikation bei ESAWN

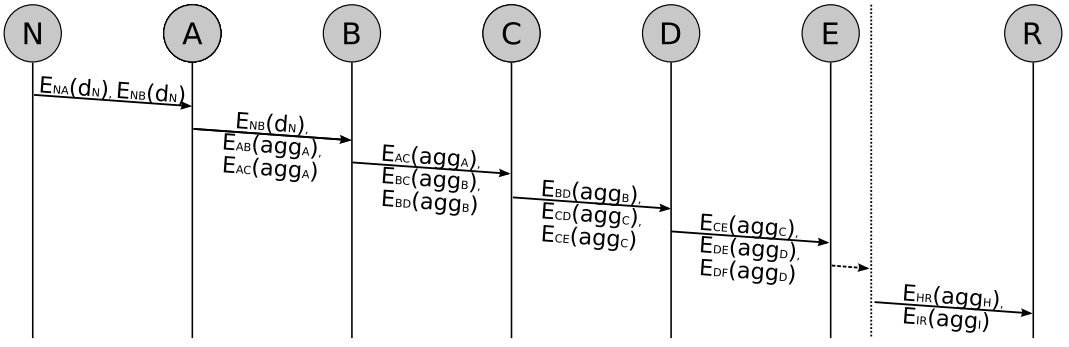


Abbildung 2.4: Visualisierung der tatsächlichen Kommunikation bei ESAWN

3. Entwurf

Ziel des Kapitels: Die aufgezeigten Probleme von ESAWN, insbesondere die fehlende Korrektheit und Non-Repudiation, sind Nachteile die, abhängig vom Anwendungsfall, nicht immer akzeptabel sind. Trotzdem hat ESAWN, gerade aufgrund der niedrigen Kosten, seine Berechtigung. Die nachfolgend vorgestellten Protokolle ESAWN-2 und ESAWN-NR sollen ESAWN daher nicht ersetzen, sondern sind Erweiterungen, die abhängig vom Anwendungsszenario ausgewählt und eingesetzt werden können.

- Der erste Protokollvorschlag, ESAWN-2, behebt das Problem des im Betrugsfall abgebrochenen Aggregationsvorgangs und verhindert außerdem eine mögliche Fehlalarmierung durch den Angreifer.
- Der zweite Protokollvorschlag, ESAWN-NR, führt die Garantien der Non-Repudiation, also der Nicht-Abstreitbarkeit von versandten Datenpaketen, mit dem ESAWN-2 Protokoll zusammen und ermöglicht damit einen korrumpierten Knoten zweifelsfrei zu identifizieren und dies den anderen legitimen Knoten mitzuteilen.

Aufbau des Kapitels: Zuerst wird in Abschnitt 3.1 ESAWN-2 vorgestellt. Der Vorstellung der Protokollidee (3.1.1) folgt eine detaillierte Darstellung des Protokolls (3.1.2). Die erzielten Protokolleigenschaften werden beschrieben (3.1.3) und anschließend bewiesen (3.1.4). Den Abschluss macht eine Protokollerweiterung die Energie einspart (3.1.5). Es folgt die Beschreibung von ESAWN-NR in Abschnitt 3.2. Nach einer Einführung in das Funktionsprinzip (3.2.1) folgt eine detaillierte Beschreibung des Protokolls (3.2.1). Anschließend werden die erzielten Protokolleigenschaften bewiesen (3.2.2). Den Abschluss macht auch hier eine Protokollerweiterung die Energie einspart (3.2.3).

3.1 ESAWN-2 – vollständig und korrekt

In seiner bisherigen Form bietet ESAWN lediglich Vollständigkeit. Dies bedeutet, dass jeder Betrugsversuch von maximal k hintereinanderliegenden korrumpierten

Knoten erkannt wird und in einer Alarmmeldung resultiert. ESAWN bietet jedoch keine Korrektheit. Knoten, die eine Alarmmeldung empfangen, können nicht erkennen, ob wirklich ein Betrug stattfand oder ob es sich um eine Falschmeldung eines korrumpierten Zeugen handelt, der den Normalbetrieb im Netzwerk stören möchte.

Dieser Tatsache lässt sich durch eine Veränderung des Protokolls begegnen, die in ESAWN-2 realisiert ist. Bevor die Protokollidee von ESAWN-2 beschrieben wird, muss jedoch klargestellt werden, dass das Erreichen von Korrektheit bei ESAWN-2 mit einem höheren Energieverbrauch einher gehen wird. Aufgrund der Randbedingungen eines drahtlosen Sensornetz (siehe 2.3), ist dies unausweichlich. Hier kann nicht auf spezielle, unterstützende Infrastrukturen, wie in einem konventionellen Netzwerk, zurückgegriffen werden kann. Mit welchem Energieverbrauch konkret zu rechnen ist, wird die Evaluation in Kapitel 5 zeigen.

3.1.1 Redundante Berechnung und Mehrheitsentscheide

ESAWN-2 realisiert seine Robustheit gegen Betrugsversuche durch das zusätzliche Durchführen von Aggregationen, ähnlich wie dies bereits bei ESAWN der Fall ist. Im Gegensatz zu ESAWN führt eine Diskrepanz zwischen diesen Aggregaten jedoch nicht zu einer Alarmmeldung, sondern der Knoten arbeitet mit dem Wert weiter, der bei dieser Berechnung mehrheitlich vorkommt.

Um solch einen Mehrheitsentscheid zu ermöglichen, müssen bei gleicher Angreiferzahl mehr zusätzliche Aggregationen durchgeführt werden, als dies bei ESAWN der Fall ist: Zum Erkennen eines Betruges ist bereits *ein* korrektes Aggregat aus der Menge aller berechneten Aggregate ausreichend. Es unterscheidet sich im Inhalt von den anderen, gefälschten, Aggregaten und führt zu einer Alarmmeldung. Für ESAWN reichen daher in Gegenwart von k korrumpierten Knoten auch k Zeugen. Dies führt zur Berechnung von $k + 1$ Aggregaten, von denen die k korrumpierten Knoten nur höchstens k fälschen können. Ein Betrugsversuch wird aufgedeckt.

Ein Mehrheitsentscheid führt in dieser Situation jedoch nicht zum Erfolg. Statt dessen muss sichergestellt werden, dass immer mehr als die Hälfte aller Aggregate den korrekten Wert repräsentieren. Da k korrumpierte Knoten k Aggregate fälschen können, müssen hier mindestens $2k$ Zeugen zum Einsatz kommen. Damit ergeben sich $2k + 1$ Aggregate von denen die korrumpierten Knoten genau k, und damit weniger als die Hälfte, fälschen können. Der $2k + 1$-Vorgänger (Entscheider) des Aggregationsknoten akzeptiert als Aggregat nur den Wert, der von mehr als der Hälfte der Knoten vertreten wird.

Solch ein Mehrheitsentscheid ist genau dann sicher, wenn aus der Menge der $2k + 1$ Aggregationsknoten höchstens k Knoten korrumpiert sind. Diese können maximal $k < k + 1$ Aggregate fälschen und erreichen nicht die $k + 1$ Aggregate, die nötig wären, dass sie der Entscheider als authentisches Aggregat akzeptiert.

Während ESAWN durch den Systemparameter k ausdrückt, gegen wieviel *hintereinanderliegende korrumpierte* Sensorknoten im Aggregationsbaum das Protokoll sicher ist, ist bei ESAWN-2 die Zahl der korrumpierten Sensorknoten in der *Menge* der $2k+1$ hintereinanderliegenden Sensorknoten maßgeblich. Um diesen Unterschied hervorzuheben, wird ab sofort im Zusammenhang mit ESAWN-2 und später auch ESAWN-NR von dem Systemparameter $\bar{k}$ die Rede sein. Dadurch wird er von dem bei ESAWN verwendeten Parameter k abgegrenzt.

Die Evaluation wird später zeigen, dass aus k und $\bar{k}$ tatsächlich unterschiedliche Sicherheitsgarantien hervorgehen und deshalb nicht zwangsläufig $k = \bar{k}$ gesetzt werden darf.

ESAWN-2 setzt generell „authenticated encryption“ [30] ein. Damit wird nicht nur Vertraulichkeit durch Verschlüsselung erreicht, sondern auch die Authentizität der Daten bezüglich des Ursprungs und die Unversehrtheit der Daten sichergestellt. Dies wird durch den Anhang einer Prüfsumme, oder ähnlichem, an die Nutzdaten erreicht. Beides zusammen mit einem symmetrischen Schlüssel, den nur die beiden kommunizierenden Sensorknoten kennen, verschlüsselt. Eine Manipulation durch einen Dritten fällt beim Entschlüsseln durch falsche Prüfdaten auf. Stimmen die Prüfdaten dagegen, kann sich der Empfänger sicher sein, unveränderte Daten erhalten zu haben, die vom Sender stammen.

3.1.2 ESAWN-2 im Detail

Die nachfolgende Beschreibung stellt ESAWN-2 im einzelnen vor. Der Parameter $\bar{k}$ gibt dabei die Zahl der Knoten an die in einer Menge von $2\bar{k} + 1$ hintereinanderliegenden Knoten maximal korrumpiert sein dürfen, ohne dass ein Betrug möglich ist.

- Jeder Blattknoten N sendet seinen Messwert d_N an seinen direkten Vorgänger und an den 2-Vorgänger, ..., $(2\bar{k} + 1)$-Vorgänger im Aggregationsbaum. Die Daten verschlüsselt N jeweils direkt für den jeweiligen Empfänger. N versendet damit d_N insgesamt $2\bar{k} + 1$ mal, immer unterschiedlich verschlüsselt.
- Jeder Aggregationsknoten A_i berechnet sein eigenes Aggregat agg_{A_i} sowie das seiner direkten und darauf folgenden Kindknoten, bis zu einer Tiefe von $2\bar{k}$, falls möglich. Zur Berechnung aller Aggregate verwendet der Knoten nur „korrekte“ Werte. Dabei gilt:
  - Ein von einem Blattknoten direkt empfangener Wert ist „korrekt“.
  - Ein von einem Knoten empfangenes Aggregat ist nur dann „korrekt“, wenn der Versender für dieses Aggregat zuständig ist und es auch von mindestens $\bar{k}$ anderen zuständigen Knoten empfangen wurde. Dies bedeutet, dass A_i hier einen Mehrheitsentscheid über alle $2\bar{k} + 1$ berechneten Aggregate durchführt und dabei den mehrheitlich vorkommenden Wert bestimmt. Dieser ist der „korrekte“ Wert.
- Sobald der Knoten alle Aggregate berechnet hat, die dieser im Rahmen seiner Aufgabe als Aggregator und Zeuge berechnen muss, werden diese, zusammen mit eventuell von Kindknoten weiterzuleitenden Daten, an den Vaterknoten gesendet. Viele dieser Daten muss auch der Vaterknoten lediglich weiterleiten. Im Einzelnen gilt für A_i:
  - Empfänger des Aggregats agg_{A_i} ist der $(2\bar{k} + 1)$-Vorgänger von A_i im Aggregationsbaum (der Entscheiderknoten von A_i).
  - Ein in der Funktion als Zeuge berechnetes Aggregat agg_{A_j}, dessen Aggregationsknoten A_j einen Abstand i zu A_i besitzt, ist für den $(2\bar{k} + 1 - i)$-Vorgänger bestimmt.

3.1.2.1 Beispiel mit $\bar{k} = 1$

Als erstes Beispiel dient Abbildung 3.1 mit einem stark vereinfachten Aggregationsbaum ($\delta = 1$) und $\bar{k} = 1$.

Die Notation agg_X bezeichnet dabei das authentische Aggregat an der Stelle des Knotens X. Wenn das Aggregat durch X selbst berechnet wurde, wird dies durch agg_X^0 ausgedrückt. Wurde das Aggregat durch einen der beiden Zeugen berechnet, wird dies durch agg_X^1 und agg_X^2 ausgedrückt.

Blattknoten N

N ist der Blattknoten, der seinen Messwert d_N für seine drei Vorgänger, A, B und C verschlüsselt und an seinen direkten Vorgänger A sendet.

$N \rightarrow A : E_{NA}(d_N)$, $E_{NB}(d_N)$, $E_{NC}(d_N)$.

Aggregationsknoten A

A entschlüsselt das für ihn bestimmte Datum $E_{NA}(d_N)$ und leitet die anderen Daten ($E_{NB}(d_N)$, $E_{NC}(d_N)$) später, zusammen mit den eigenen Daten, weiter. A bildet aus dem entschlüsselten Wert d_N sein Aggregat agg_A^0 und verschlüsselt dies für den Empfänger, Knoten D. Die Knoten B und C benötigen agg_A^0 nicht, denn sie können das Aggregat selbst berechnen, da sie den Wert d_N direkt von N erhalten.

$A \rightarrow B : E_{NB}(d_N)$, $E_{NC}(d_N)$, $E_{AD}(\text{agg}_A^0)$.

Aggregationsknoten B

B entschlüsselt, wie A, das für ihn bestimmte Datum $E_{NB}(d_N)$ und leitet die restlichen Daten ($E_{NC}(d_N)$, $E_{AD}(\text{agg}_A^0)$) später weiter. Als Zeuge von A muss B nun zuerst agg_A^1 berechnen und anschließend sein eigenes Aggregat agg_B^0. Die beiden Ergebnisse werden für ihre Empfänger, Knoten D und Knoten E verschlüsselt und zusammen mit den weiterzuleitenden Daten an C versendet.

$B \rightarrow C : E_{NC}(d_N)$, $E_{AD}(\text{agg}_A^0)$, $E_{BD}(\text{agg}_A^1)$, $E_{BE}(\text{agg}_B^0)$.

Aggregationsknoten C

C entschlüsselt, wie A, das für ihn bestimmte Datum $E_{NC}(d_N)$ und leitet die restlichen Daten ($E_{AD}(\text{agg}_A^0)$, $E_{BD}(\text{agg}_A^1)$, $E_{BE}(\text{agg}_B^0)$) später weiter. Als Zeuge von A und B muss C nun zuerst agg_A^2 und agg_B^1 berechnen und anschließend sein eigenes Aggregat agg_C^0. Die drei Aggregate werden für ihre Empfänger D, E und F verschlüsselt und alles an D versendet.

$C \rightarrow D : E_{AD}(\text{agg}_A^0)$, $E_{BD}(\text{agg}_A^1)$, $E_{BE}(\text{agg}_B^0)$, $E_{CD}(\text{agg}_A^2)$, $E_{CE}(\text{agg}_B^1)$, $E_{CF}(\text{agg}_C^0)$

Aggregationsknoten D

Knoten D ist der erste Aggregationsknoten, der keinen direkten Zugriff mehr auf das Messdatum d_N besitzt. Er muss sein Aggregat agg_D^0 ohne diese Hilfe aus dem Aggregat agg_A berechnen. Dieses Aggregat bestimmt D durch einen Mehrheitsentscheid über den drei empfangenen Daten agg_A^0, agg_A^1 und agg_A^2. D stellt so sicher, dass er mit dem korrekten Wert für agg_A arbeitet. Damit kann D nun seine Aggregate agg_B^2, agg_C^1 und agg_D^0 bestimmen, die D als Zeuge beziehungsweise Aggregator berechnen muss. Alle berechneten und weiterzuleitenden Daten werden anschließend an E versendet.

$D \rightarrow E : E_{BE}(\text{agg}_B^0)$, $E_{CE}(\text{agg}_B^1)$, $E_{CF}(\text{agg}_C^0)$, $E_{DE}(\text{agg}_B^2)$, $E_{DF}(\text{agg}_C^1)$, $E_{DG}(\text{agg}_D^0)$

Aggregationsknoten E

Wie Knoten D muss auch Knoten E durch einen Mehrheitsentscheid die Korrektheit der Eingangsdaten sicherstellen. Dazu bestimmt er agg_B durch Mehrheitsentscheid über agg_B^0, agg_B^1 und agg_B^2. Anschließend berechnet E als Zeuge agg_C^2 und agg_D^1 sowie als Aggregator sein Aggregat agg_E^0. Alle Daten werden danach an F versendet.

$E \rightarrow F : E_{CF}(\text{agg}_C^0)$, $E_{DF}(\text{agg}_C^1)$, $E_{DG}(\text{agg}_D^0)$, $E_{EF}(\text{agg}_C^2)$, $E_{EG}(\text{agg}_D^1)$, $E_{EH}(\text{agg}_E^0)$.

Aggregationsknoten F und G

F und G verfahren entsprechend Knoten E. Zuerst der Mehrheitsentscheid, um die Aggregate agg_D und agg_E zu bestimmen. Anschließend die Aggregatsberechnungen und das Versenden aller Daten an G und H.

$F \rightarrow G : E_{DG}(\text{agg}_D^0)$,$E_{EG}(\text{agg}_D^1)$, $E_{EH}(\text{agg}_E^0)$, $E_{FG}(\text{agg}_D^2)$, $E_{FH}(\text{agg}_E^1)$, $E_{FI}(\text{agg}_F^0)$.

$G \rightarrow H : E_{EH}(\text{agg}_E^0)$, $E_{FH}(\text{agg}_E^1)$, $E_{FI}(\text{agg}_F^0)$, $E_{GH}(\text{agg}_E^2)$, $E_{GI}(\text{agg}_F^1)$, $E_{GR}(\text{agg}_G^0)$.

Aggregationsknoten H

Das Aggregat agg_H^0 von Knoten H wird für den restlichen Verlauf der Aggregation nicht benötigt, da es zwischen H und R nicht mehr genügend Knoten gibt, um die Aussage von H über sein Aggregat durch einen Mehrheitsentscheid zu überprüfen. Deshalb versendet H neben den weiterzuleitenden Daten nur die von ihm als Zeuge berechneten Aggregate agg_F^2 und agg_G^1.

$H \rightarrow I : E_{FI}(\text{agg}_F^0)$, $E_{GI}(\text{agg}_F^1)$, $E_{GR}(\text{agg}_G^0)$, $E_{HI}(\text{agg}_F^2)$, $E_{HR}(\text{agg}_G^1)$.

Aggregationsknoten I

Auch das Aggregat agg_I^0 von Knoten I wird für die restliche Aggregation nicht benötigt. Deshalb versendet I neben den weiterzuleitenden Daten nur das Aggregat agg_G^2, um den Mehrheitsentscheid über agg_G von Knoten R zu ermöglichen.

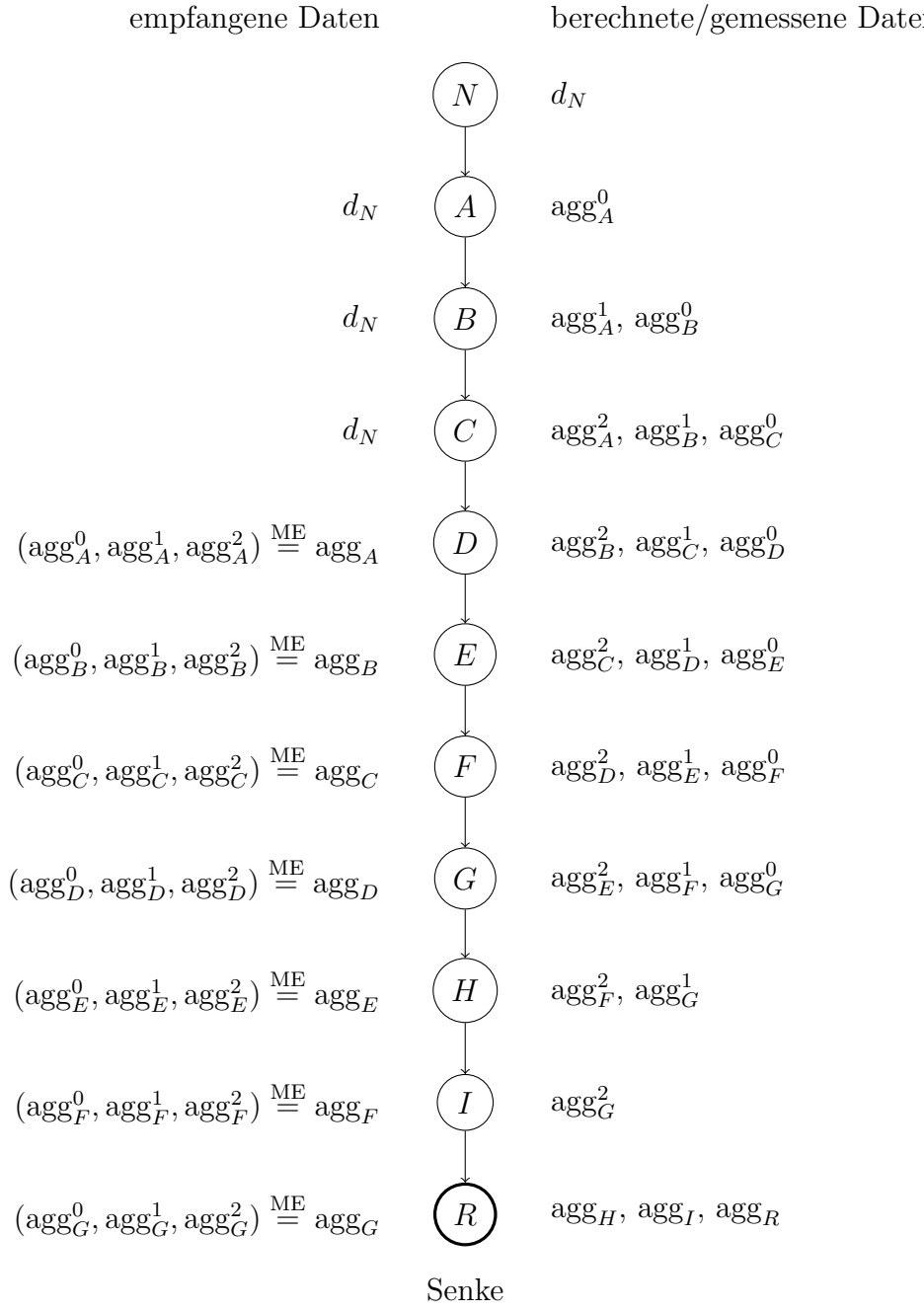
$I \rightarrow R : E_{GR}(\text{agg}_G^0)$, $E_{HR}(\text{agg}_G^1)$, $E_{IR}(\text{agg}_G^2)$.

Senke R

Die Senke R kann dann durch Mehrheitsentscheid agg_G bestimmen und daraus agg_H, agg_I und agg_R, das letzte Aggregat und Gesamtergebnis der Aggregation, berechnen.

Ablaufdiagramme

Zwei Ablaufdiagramme, in den Abbildungen 3.2 und 3.3, zeigen die Kommunikation der Knoten dieses Netzwerks detailliert. In der ersten Abbildung 3.2 wird die theoretische Kommunikation dargestellt, also die logischen Kommunikationsverbindungen zwischen zwei Knoten. In der zweiten Abbildung 3.3 ist die tatsächliche Kommunikation ersichtlich. Diese unterscheidet sich deutlich von der ersten Darstellung. Viele der Daten können zusammen verschickt werden und werden daher von den Knoten zusammengefasst und in einer Übertragung versandt. Diese Paketaggregation ermöglicht zusätzliche Energieeinsparungen. Abhängig von der maximalen Nutzlast der Pakete kann die Zahl der Pakete so bis auf eins reduziert werden.



d_N:	Messwert von Knoten N
agg_A^0:	Aggregat A, durch A berechnet
agg_A^i:	Aggregat A, durch den i-ten Zeugen von A berechnet
agg_A:	Aggregat A, durch Mehrheitsentscheid bestimmt
$\overset{\text{ME}}{=}$:	Durchführen des Mehrheitsentscheids

Abbildung 3.1: Protokollablauf bei ESAWN-2, $\delta = 1$ und $\bar{k} = 1$

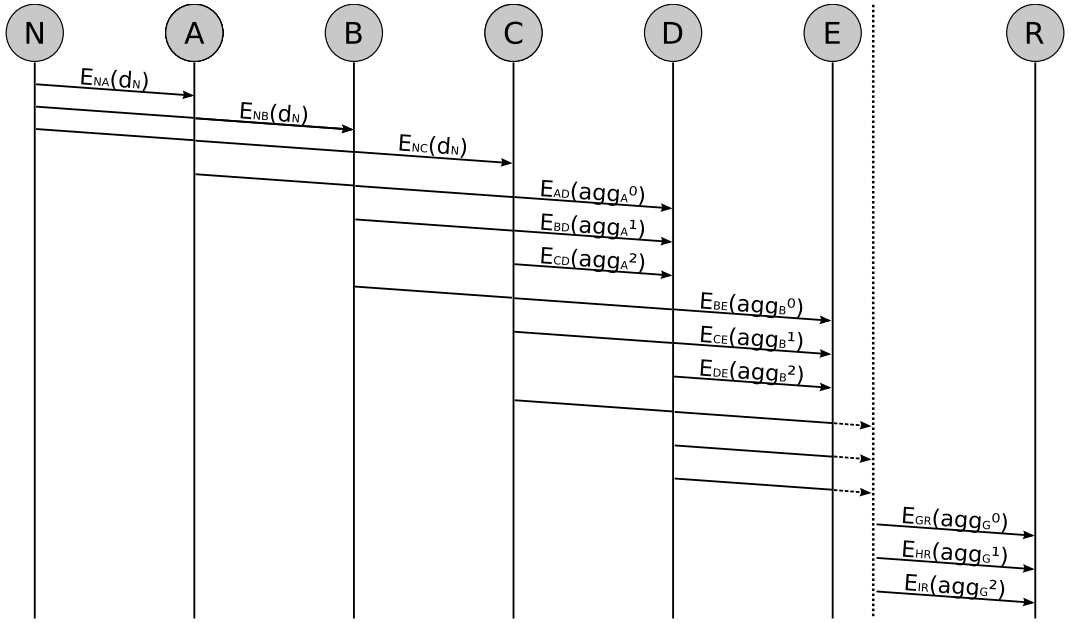


Abbildung 3.2: Visualisierung der theoretischen Kommunikation bei ESAWN-2

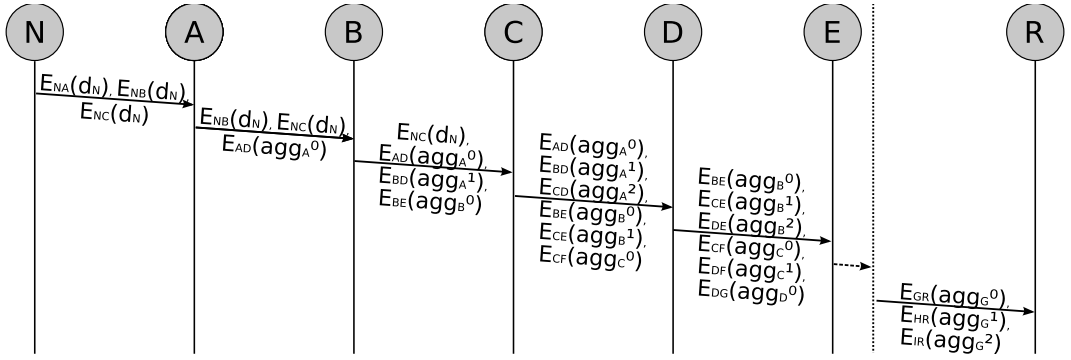


Abbildung 3.3: Visualisierung der tatsächlichen Kommunikation bei ESAWN-2

3.1.2.2 Beispiel mit $\bar{k} = 2$

Als zweites Beispiel dient Abbildung 3.4, die den gleichen Baum zugrunde legt, jedoch mit $\bar{k} = 2$. Dies bedeutet, dass ein Mehrheitsentscheid auf fünf statt bisher drei Aggregaten aufbaut. Deshalb muss der Blattknoten N seinen Messwert d_N auch an mehr Knoten entlang des Aggregationspfades schicken um dies zu ermöglichen. Dadurch steigt die zu übertragende Datenmenge und die Zahl der durchzuführenden Aggregationen deutlich, was zu höheren Energiekosten führt.

3.1.3 Korrektheit und Vollständigkeit

ESAWN-2 bietet Korrektheit und Vollständigkeit, wenn $\bar{k}$ entsprechend der Zahl und Verteilung der korrumpierten Knoten gewählt ist. Für alle $2\bar{k}$ hintereinanderliegenden Knoten im Aggregationsbaum gilt, dass höchsten $\bar{k}$ von ihnen korrumpiert sind. Dadurch ist eine Mehrheit an korrekten Aggregaten für jeden Mehrheitsentscheid im Baum sichergestellt. Ein Betrugsversuch hat keinen Einfluss auf den Ausgang der Aggregation. Diese Tatsache wird in Abschnitt 3.1.4 formal bewiesen.

Somit kann die Gesamtzahl der korrumpierten Knoten im Netzwerk durchaus höher als $\bar{k}$ liegen, vorausgesetzt die Knoten sind so verteilt, dass sie die genannte Bedingung nicht verletzen. Ein Betrug ist erst dann erfolgreich, wenn mehr als die Hälfte der Aggregate gefälscht werden kann. Es sind mindestens $\bar{k}+1$ korrumpierte Knoten notwendig, die sich auf insgesamt $2\bar{k}+1$ hintereinanderliegende Aggregationsknoten verteilen müssen. Sonst können die korrumpierten Knoten nicht gemeinsam Einfluss auf die gleiche Mehrheitsentscheidung ausüben.

Im Beispiel aus Abbildung 3.1 gilt, dass durch $\bar{k} = 1$ höchstens ein Knoten aus drei hintereinanderliegenden Knoten korrumpiert sein darf, um die Korrektheit der Aggregation nicht zu gefährden. Sind es zwei korrumpierte Knoten, dürfen diese nie Teil des selben Mehrheitsentscheids werden. Dies bedeutet bei $\bar{k} = 1$, dass zwischen den beiden Knoten mindestens zwei legitime Knoten auf dem Aggregationspfad liegen müssen. Sind beispielsweise die Knoten C und F korrumpiert, können diese nicht den Ausgang der Aggregation beeinflussen. C hat alleine zu geringen Einfluss auf die Mehrheitsentscheide der Knoten D und E. F wiederum zu geringen Einfluss bei Knoten G, H und I.

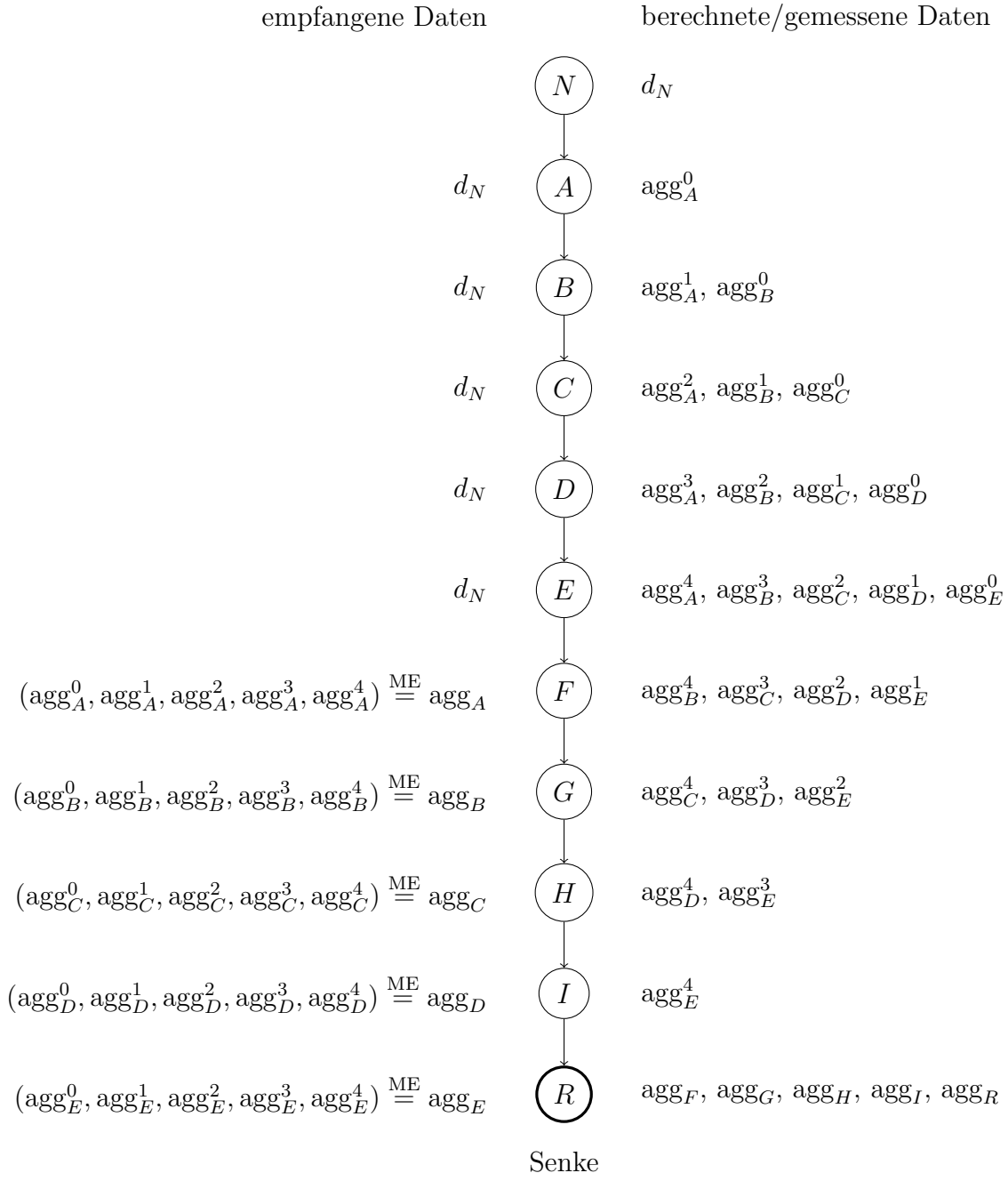
Im Vergleich dazu das Beispiel aus Abbildung 3.4. Da hier $\bar{k} = 2$ zum Einsatz kommt, können auch zwei Angreifer Teil des gleichen Mehrheitsentscheides werden, ohne mit einem Betrugsversuch erfolgreich zu sein.

3.1.4 Beweis der Korrektheit und Vollständigkeit

Im Folgenden wird bewiesen, dass ESAWN-2 die geforderten Eigenschaften Vollständigkeit und Korrektheit tatsächlich bietet. Dazu wird gezeigt, dass ESAWN-2 für eine gewisse Zahl und Verteilung der Angreifer fehlerfrei arbeitet.

ESAWN-2 läuft fehlerfrei, wenn unabhängig von der Präsenz korrumpierter Knoten, jeder legitime Aggregationsknoten sein authentisches Aggregat kennt. Der Wert des authentischen Aggregats entspricht dem Wert, den das Aggregat im selben Netzwerk ohne korrumpierte Knoten hat.

Der Beweis ist in Anlehnung an den Beweis für ESAWN in [6] entstanden und nutzt, wenn möglich, die gleichen Definitionen.



d_N:	Messwert von Knoten N
agg_A^0:	Aggregat A, durch A berechnet
agg_A^i:	Aggregat A, durch den i-ten Zeugen von A berechnet
agg_A:	Aggregat A, durch Mehrheitsentscheid bestimmt
$\stackrel{\mathrm{ME}}{=}$:	Durchführen des Mehrheitsentscheids

Abbildung 3.4: Protokollablauf bei ESAWN-2, $\delta = 1$ und $\bar{k} = 2$

3.1.4.1 Voraussetzungen

Der Beweis verwendet die folgenden Definitionen und den Parameter $\bar{k}$ sowie eine im Folgenden beschriebene Farbfunktion, die als gegeben betrachtet werden.

Gegeben

- Parameter $\bar{k}$
- Eine **Farbfunktion** $c : V \rightarrow \{\text{schwarz}, \text{weiß}\}$ die jedem Knoten aus V eines Graphen die Farbe weiß oder schwarz zuweist.

Definitionen

- Ein **Baum** $B = (V, E)$ ist ein zyklenfreier, gerichteter Graph mit einer Menge von Knoten (V) und einer Menge gerichteter Kanten (E), so dass jeder Knoten genau einen **Vorgänger** mit seiner ausgehenden Kante bestimmt. Ausnahme ist die Wurzel w die keinen Vorgänger besitzt. Knoten ohne eingehende Kanten sind **Blattknoten**.
- Ein **Pfad** $\mathcal{P}$ in B ist eine Folge von Knoten $\mathcal{P} = v_1, v_2, \ldots, v_l$ wobei $v_i \in V$ ist und v_i's ausgehende Kante auf den Vorgänger v_{i+1} zeigt. Somit liegen die l Knoten $v_1, v_2, \ldots, v_l$ **hintereinander** auf $\mathcal{P}$
- Ein **Blatt-Pfad** $\mathcal{P}_B$ ist ein Pfad $\mathcal{P}$ in einem Baum B der bei einem Blattknoten beginnt.
- Der **1-Vorgänger** eines Knotens v ist der Vorgänger von v.
- Der **r-Vorgänger** eines Knotens v ist der Vorgänger des $(r-1)$-Vorgängers.
- Unter dem $(2\bar{k}+1)$**-Vorgänger-Abschluss** B^+ von B versteht man einen gerichteten Graphen, der B enthält und zusätzlich Kanten von jedem Knoten zu seinem 1-Vorgänger, 2-Vorgänger, …, $(2\bar{k}+1)$-Vorgänger, soweit es diese Vorgänger in B gibt.
- Der **weiße Teilgraph** eines mit c gefärbten Graphen ist derjenige Teilgraph von G der nur die weißen Knoten sowie deren Kanten untereinander enthält.
- Der **Grad** δ eines Baumes B gibt die maximale Anzahl eingehender Kanten jedes Knotens $v_i \in V$ an.
- Ein Graph ist **ausreichend verbunden**, wenn für jeden Knoten K_i und jeden Blatt-Pfad der bei K_i endet gilt: K_i besitzt mindestens $\bar{k}+1$ eingehende Kanten von anderen Aggregationsknoten *oder* eine eingehende Kante von dem Blattknoten auf diesem Pfad *oder* K_i ist selbst Blattknoten.
- Ein mit c gefärbter Baum B ist **einwandfrei**, wenn der weiße Teilgraph von B^+ ausreichend verbunden ist.

3.1.4.2 Annahme über den Baumaufbau

ESAWN-2 ist nur bis zu einer gewissen Angreiferzahl und Verteilung fehlerfrei. Die Bedingung an den Aggregationsbaum wird mit der Annahme $A_{\bar{k}}$ beschrieben.

Annahme $A_{\bar{k}}$

Von beliebigen $(2\bar{k}+1)$ hintereinanderliegenden Knoten auf jedem Pfad sind mindestens $\bar{k}+1$ weiß. Alle Blattknoten sind weiß.

3.1.4.3 Beweis

Der Beweis geht in zwei Schritten vor. Im ersten Schritt wird gezeigt, dass Bäume, die die Annahme $A_{\bar{k}}$ erfüllen, einwandfreie Bäume sind. Im zweiten Schritt wird bewiesen, dass ESAWN-2 auf Sensornetzen deren Aggregationsbaum ein einwandfreier Baum ist, fehlerfrei läuft. Beide Teilschritte verbunden ergeben dann, dass ESAWN-2 auf allen Sensornetzen deren Aggregationsbaum $A_{\bar{k}}$ erfüllt, fehlerfrei läuft.

Behauptung

Bäume mit weißer Wurzel, die $A_{\bar{k}}$ erfüllen sind einwandfreie Bäume.

Strukturelle Induktion über B

1. Induktionsanfang:
   Gegeben ist ein Baum B der $A_{\bar{k}}$ erfüllt und nur einen Knoten w enthält. Gemäß Definition ist der weiße Teilgraph des $(2\bar{k}+1)$-Vorgänger-Abschlusses w selbst. Da w in diesem Fall ein Blattknoten des Baumes darstellt, ist der weiße Teilgraph von B^+ ausreichend verbunden, also der Baum einwandfrei.

2. Induktionsvoraussetzung:
   Gegeben seien maximal δ einwandfreie Bäume B_1, B_2, …, B_δ welche die Behauptung erfüllen.

3. Induktionsschluss:
   Zu zeigen ist: Für einen neuen Baum B mit der weißen Wurzel w, die mit den Wurzeln der einwandfreien Teilbäumen B_1, B_2, …, B_δ verbunden wird (siehe Abbildung 3.5), gilt: B ist ebenfalls einwandfrei.
   Beweis:
   Durch die neue Wurzel w ändern sich die Zahl und Art und Weise der eingehenden Kanten in dem weißen Teilgraphen des $(2\bar{k}+1)$-Vorgänger-Abschluss von B_i nicht. Dies bedeutet, dass die Knoten in B_i, auch als Bestandteil von B, die Kriterien eines einwandfreien Baums weiterhin erfüllen. Es bleibt zu prüfen, ob im weißen Teilgraphen von B^+ auch Knoten w ausreichend verbunden und damit der gesamte Baum B einwandfrei ist.

   Der Knoten w ist Teil des weißen Teilgraphen von B^+. Damit der gesamte Baum B einwandfrei ist, muss für alle Blattpfade, die in diesem weißen Teilgraphen bei w enden, gelten, dass w mindestens $\bar{k}+1$ eingehende Kanten von anderen Nicht-Blattknoten oder eine eingehende Kante von dem Blattknoten auf diesem Blattpfad besitzt:

   - Gilt für die Länge des Blattpfades $l : l \leq (2\bar{k}+1)$, dann besitzt w eine direkte Kante zu dem Blattknoten und die Bedingung ist erfüllt.
   - Ist der Blattpfad länger, müssen sich unter den $2\bar{k}+1$ Vorgängern von w mindestens $\bar{k}+1$ weiße Knoten befinden. Auch dies ist der Fall, denn Bedingung $A_{\bar{k}}$ fordert genau dieses.

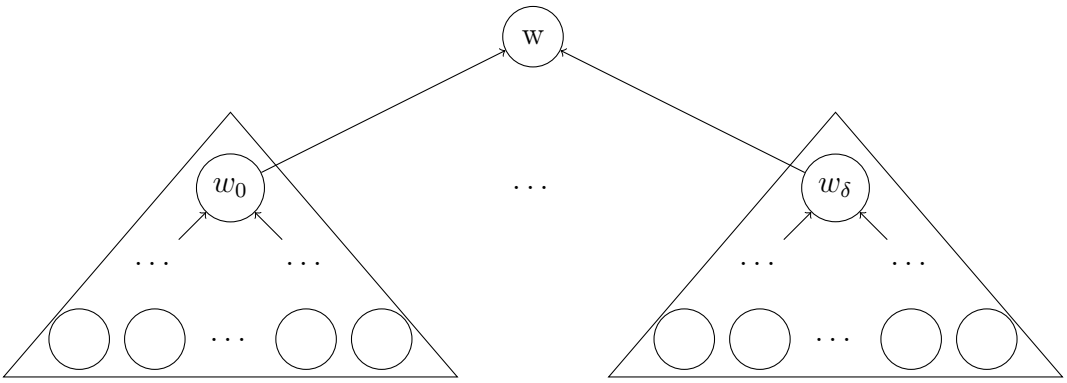


Abbildung 3.5: ESAWN-2 Beweis – Induktionsschluss

Übertragung auf ESAWN-2

In ESAWN-2 ist ein Betrugsversuch erfolgreich, wenn sich mindestens ein legitimer Aggregationsknoten im Rahmen des Mehrheitsentscheids für das falsche Aggregat entscheidet. Dies passiert genau dann, wenn sich unter den Sensorknoten die diesen Aggregationsknoten mit Daten versorgen, mehr korrumpierte als legitime Sensorknoten befinden. Gilt dagegen, dass bei allen Mehrheitsentscheidungen die Zahl der legitimen Knoten überwiegt, ist der Aggregationsvorgang fehlerfrei.

Daher zeigt dieser Beweis, dass ESAWN-2 Betrugsversuche effektiv unterbinden kann. Dies wird klar, wenn man einen beliebigen Aggregationsbaum als den Baum B aus dem Beweis betrachtet. Weiße Knoten stellen legitime Sensorknoten dar, schwarze Knoten sind korrumpierte Sensorknoten. Die Kanten stellen die Aggregationsbeziehungen zwischen den Knoten dar.

Die Annahme $A_{\bar{k}}$ besagt nun – transferiert auf das Sensornetz – dass unter $(2\bar{k}+1)$ hintereinanderliegenden Sensorknoten höchstens $\bar{k}$ Knoten korrumpiert sein dürfen. Der Beweis zeigt, dass (Aggregations-)Bäume, die diese Annahme erfüllen, einwandfreie Bäume sind. In solch einem einwandfreien Baum kann ESAWN-2 fehlerfrei arbeiten. Es ist sichergestellt, dass immer mehr als die Hälfte der Sensorknoten, die einem Knoten Daten liefern, legitim sind oder der Knoten direkt Zugriff auf die Messdaten eines Blattknotens besitzt. In einer solchen Umgebung hat kein Betrugsversuch eines korrumpierten Knotens eine Chance von einem legitimen Knoten akzeptiert zu werden.

3.1.5 Einführung einer Prüfwahrscheinlichkeit p für Aggregationsknoten

Durch die Verwendung von Zeugen zur zusätzlichen Berechnung von Aggregaten steigt die Zahl der Nachrichten und damit die Energiekosten deutlich. ESAWN kennt daher mit dem Parameter p eine Möglichkeit die Energiekosten zu reduzieren, in dem ein Aggregationsknoten nicht immer durch Zeugen überprüft wird. Der Parameter wird, genau wie $\bar{k}$, durch den Nutzer des Protokolls vorab festgelegt. Das detaillierte Vorgehen dieses Mechanismus wurde in meiner Studienarbeit [41] beschrieben.

Diese Idee lässt sich auf ESAWN-2 übertragen. p legt hier fest, mit welcher Wahrscheinlichkeit ein Aggregationsknoten durch einen Mehrheitsentscheid kontrolliert

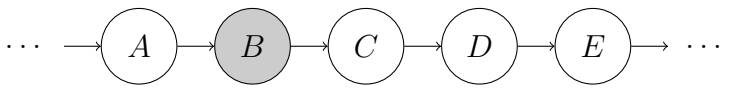


Abbildung 3.6: Auszug aus einem Aggregationspfad, $\bar{k} = 1$

wird. Wird ein Knoten überprüft, akzeptiert sein $(2\bar{k} + 1)$-Vorgänger als Aggregat nur den Wert, der im Rahmen der Berechnung durch die Zeugen mehrheitlich vorkommt. Wird ein Knoten nicht überprüft, akzeptiert sein $(2\bar{k} + 1)$-Vorgänger das Aggregat direkt, als ob es von einem Blattknoten stammen würde.

Um das Verfahren der Prüfwahrscheinlichkeit effektiv nutzen zu können, ist die Synchronisation der Zeugen und des Entscheiderknotens notwendig. Alle beteiligten Knoten müssen zu dem selben Schluss hinsichtlich einer Überprüfung des Aggregationsknotens kommen. Um dies zu erreichen, wird auf den aus ESAWN bekannten Pseudo-Zufallszahlen-Mechanismus zurückgegriffen.

Eine Reduzierung der Prüfwahrscheinlichkeit auf $p < 1$ hat Einfluss auf die WKA. Sobald ein Knoten nicht mehr bei jeder Aggregation überprüft wird, steigt die Wahrscheinlichkeit für einen Angreifer erfolgreich einen Betrugsversuch durchzuführen. Der genaue Einfluss von p auf die WKA ist in Abschnitt 5.4.1.1 der Evaluation beschrieben.

3.2 ESAWN-NR: ESAWN-2 mit Non-Repudiation

Durch die Verwendung von Mehrheitsentscheiden ist ESAWN-2 in der Lage korrumpierte Daten zu erkennen und mit den korrekten Daten weiterzuarbeiten. Auch wenn der Knoten, der den Mehrheitsentscheid durchführt, sich sehr wohl bewusst ist, wer ihm korrumpierte Daten geschickt hat (da „authenticated encryption“ zum Einsatz kommt), kann er diese Information nicht an andere legitime Knoten weitergeben. Aufgrund des Zuordnungsproblems, das beim Einsatz symmetrischer Kryptographie besteht, können sie dieser Information nicht vertrauen. So ist ein Ausschluss des korrumpierten Knotens aus dem Netzwerk nicht möglich.

Dies wird auch an folgendem Beispiel in Abbildung 3.6 deutlich. Dabei sei A der Aggregationsknoten, B und C seien beide Zeugen und D der Knoten der auf allen drei Werten den Mehrheitsentscheid durchführt. B sei nun ein korrumpierter Knoten. Da alle drei Knoten ihr Aggregat an D schicken und dabei ihren individuellen Schlüssel mit D verwenden kann D feststellen, dass B ihm andere Daten als A und C geschickt haben. Doch kann er die anderen legitimen Knoten nicht davon überzeugen, weder einen der zuständigen legitimen Knoten, zum Beispiel A, noch einem der nachfolgenden Knoten, beispielsweise E. Er hat keinen Beweis dafür, dass B wirklich falsche Daten gesendet hat, denn das empfangene Paket $E_{BD}(\text{agg})$ könnte von D selbst manipuliert worden sein, um B fälschlich anzuzeigen.

Diesem Umstand kann durch die Realisierung von „Non-Repudiation“ begegnet werden. Wie schon bei ESAWN-2 muss auch hier klargestellt werden, dass diese zusätzliche Anforderung an das Protokoll zur authentischen Aggregation, mit einem nennenswerten Mehrverbrauch an Energie verbunden sein wird.

Für „Non-Repudiation“ gibt es Methoden, die auf asymmetrischer und auf symmetrischer Kryptographie aufbauen [22]. Da auf asymmetrische Kryptographie verzichtet

werden soll, bleiben nur die Methoden, die auf symmetrischer Kryptographie aufbauen [21]. „Non-Repudiation“ ist so nur sehr aufwendig zu realisieren, etwa durch eine vertrauenswürdige dritte Instanz, einen „Notar“. Dies ist in einem drahtlosen Sensornetz jedoch nicht möglich, da von solch einem Vertrauen nicht ausgegangen werden kann. ESAWN-NR greift daher einen anderen, teureren Ansatz zur Realisierung auf. Jeder der zuständigen Knoten eines Aggregationsschrittes kann im Falle eines Betrugsversuch selbst nachprüfen, mit welchen Daten die anderen zuständigen Knoten zu der Aggregation beigetragen haben.

Wie schon ESAWN-2, greift auch ESAWN-NR zum Ver- und Entschlüsseln der Kommunikation auf „authenticated encryption“ zurück, um Vertraulichkeit und Authentizität der Daten sicherzustellen. Auch das Prinzip der Mehrheitsentscheide über insgesamt $2\bar{k}+1$ berechnete Aggregate bleibt unverändert.

3.2.1 ESAWN-NR im Detail

Im Folgenden ist es das Ziel, ESAWN-2 so zu erweitern, dass alle für einen Aggregationsschritt zuständigen Knoten von einem Betrug erfahren und diese Information auch an andere Knoten weitergeben können. Dazu wird zuerst der Weg, über den die Datenpakete ihren Empfänger erreichen, so geändert, dass alle zuständigen Knoten die versendeten Pakete der anderen zuständigen Knoten protokollieren können. Dies bedeutet, dass jeder der zuständigen Knoten speichert, welche Daten die anderen Knoten an den Entscheiderknoten gesendet haben.

Dazu senden alle Zeugen ihr Datenpaket zuerst an den Aggregationsknoten, und dieser schickt die Pakete zusammen mit seinem eigenen Aggregat zum Entscheider. Jeder Knoten den das Paket dann auf seinem Weg zum Entscheider passiert, verschlüsselt das Paket zusätzlich mit dem Schlüssel den dieser Knoten mit dem Entscheider gemeinsam hat. Dadurch wird verhindert, dass ein Datenpaket, das ein korrumpierter Knoten versendet hat, auf seinem Weg zum Entscheider von dem korrumpierten Knoten selbst oder einem anderen korrumpierten Knoten modifiziert wird und so Knoten unterschiedliche Daten protokollieren.

Ausserdem muss jedes Datenpaket vor der Übertragung zwischen zwei Knoten durch deren gemeinsamen Schlüssel geschützt werden. Damit kann der Empfänger den Ursprung des Paketes sicherstellen. Möchte beispielsweise Knoten A an Knoten B das Datenpaket $E_{AC}(agg_A)$ senden, so übertrag er $E_{AB}(E_{AC}(agg_A))$. Würde auf diese zusätzliche Verschlüsselung verzichtet, könnte B nicht sicherstellen, ob er $E_{AC}(agg_A)$ von A oder C erhalten hat.

Am Beispiel aus Abbildung 3.6 wird der neue Kommunikationsablauf ersichtlich:

Zu Beginn liegen den Knoten A, B und C die Daten aus den vorhergehenden Aggregationsschritten vor, da die einzelnen Aggregationsschritte, wie schon bei ESAWN-2, induktiv aufeinander aufbauen. Dies bedeutet, dass alle drei Knoten jetzt das Aggregat von Knoten A berechnen und für den Empfängerknoten D verschlüsseln können. Die Zeugen B und C schicken jetzt ihr verschlüsseltes Paket aber nicht direkt an D sondern an A:

$C \rightarrow B \quad E_{BC}(E_{CD}(\mathrm{agg}_A^2))$
$B \rightarrow A \quad E_{AB}(E_{BD}(\mathrm{agg}_A^1, E_{CD}(\mathrm{agg}_A^2)))$

A sendet diese beiden Datenpakete zusammen mit seinem eigenen Aggregat agg_A^0 über B und C an D:

$A \to B \quad E_{AB}(E_{AD}(\text{agg}_A^0, E_{BD}(\text{agg}_A^1, E_{CD}(\text{agg}_A^2))))$
$B \to C \quad E_{BC}(E_{BD}(E_{AD}(\text{agg}_A^0, E_{BD}(\text{agg}_A^1, E_{CD}(\text{agg}_A^2)))))$
$C \to D \quad E_{CD}(E_{BD}(E_{AD}(\text{agg}_A^0, E_{BD}(\text{agg}_A^1, E_{CD}(\text{agg}_A^2)))))$

Da die Knoten die Datenpakete protokollieren, liegt den drei Knoten A, B und C nun folgende Information im Speicher vor:

$A: \quad E_{BD}(\text{agg}_A^1, E_{CD}(\text{agg}_A^2))$
$B: \quad E_{AD}(\text{agg}_A^0, E_{BD}(\text{agg}_A^1, E_{CD}(\text{agg}_A^2)))$
$C: \quad E_{BD}(E_{AD}(\text{agg}_A^0, E_{BD}(\text{agg}_A^1, E_{CD}(\text{agg}_A^2))))$

D kann aus diesem Paket die drei Aggregate agg_A^0, agg_A^1 und agg_0^2 entschlüsseln und den Mehrheitsentscheid durchführen. Der Mehrheitsentscheid läuft dabei wie bei ESAWN-2 ab. Allerdings kann jetzt zusätzlich jedem der zuständigen Knoten nachgewiesen werden, welche Daten er an den Entscheider D gesandt hat. Im Falle eines Betrugsversuchs können dadurch alle zuständigen Knoten, aber auch alle anderen Knoten des Aggregationspfades, darüber informiert werden. Dies funktioniert in zwei Schritten:

Schritt 1

Entdeckt der legitime Entscheider im Rahmen des Mehrheitsentscheids einen Betrug, meldet er dies den zuständigen Knoten und legt als Beweis die Kommunikationsschlüssel offen, die in diesem Aggregationsschritt von ihm genutzt wurden. Damit können die Knoten die vorher protokollierten Daten entschlüsseln und die Betrugsmeldung selbst verifizieren.

Im Fall eines korrumpierten Entscheiders fällt der Betrugsversuch im nächsten darauf folgenden Aggregationsschritt mit legitimem Entscheider auf. Dies liegt am induktiven Aufbau des Protokolls, bei dem jeder der inneren Knoten in einem der Aggregationsschritte die Entscheiderfunktion übernimmt (siehe Beispiel von ESAWN-2, Abbildungen 3.1 und 3.4). Dadurch wird auch im Falle eines korrumpierten Entscheiders keinem legitimen Knoten ein korruptes Aggregat „untergeschoben“.

Ein Nachteil dieses Vorgehens ist, dass einmal offengelegte Schlüssel nicht mehr verwendet werden können. Wenn mehr als zwei Knoten den Schlüssel kennen, kann die Authentizität nicht mehr sichergestellt werden.

Schritt 2

Im zweiten Schritt werden die verbleibenden Knoten auf dem Aggregationspfad über den korrumpierten Knoten informiert. Dazu schickt jeder zuständige Knoten, der den Betrugsversuch verifiziert hat, eine Alarmmeldung an seine nächsten $2\bar{k} + 1$ Vorgänger.

Ein bei diesem Aggregationsschritt nicht zuständiger Knoten akzeptiert solch eine Alarmmeldung genau dann, wenn er sie von mindestens $\bar{k} + 1$ Knoten aus insgesamt $2\bar{k} + 1$ hintereinanderliegenden Knoten erhält. So stellt dieser sicher, dass die Alarmmeldung berechtigt ist, da davon ausgegangen wird, dass höchstens $\bar{k}$ der $2\bar{k}+1$ hintereinanderliegenden Knoten korrumpiert sind.

Hat ein Knoten die Alarmmeldung akzeptiert, schickt auch er eine entsprechende Alarmmeldung an seine nächsten $2\bar{k} + 1$ Vorgänger. So wird sichergestellt, dass die Information letztendlich die Senke erreicht.

Am Beispiel der Abbildung 3.6 lässt sich dies illustrieren. Es sind drei Fälle zu unterscheiden:

- Knoten D stellt beim Mehrheitsentscheid von agg_A fest, dass $\text{agg}_A^0 \neq \text{agg}_A^1 = \text{agg}_A^2$ gilt. Deshalb ist A korrumpiert. D meldet dies an B und C und hängt seine Schlüssel E_{AD} und E_{BD} an. Knoten B kann damit seine protokollierten Daten entschlüsseln und stellt $\text{agg}_A^0 \neq \text{agg}_A^1$ fest. Knoten C stellt $\text{agg}_A^0 \neq \text{agg}_A^2$ fest. Damit sind beide Knoten, B und C, über den Betrug informiert und melden ihre Erkenntnis an E und die folgenden Knoten auf dem Aggregationspfad.
- Knoten D stellt beim Mehrheitsentscheid von agg_A fest, dass $\text{agg}_A^1 \neq \text{agg}_A^0 = \text{agg}_A^2$ gilt. Deshalb ist B korrumpiert. D meldet dies an A und C und hängt seine Schlüssel E_{AD} und E_{BD} an. Knoten A kann damit seine protokollierten Daten entschlüsseln und stellt $\text{agg}_A^1 \neq \text{agg}_A^0$ fest. Knoten C stellt $\text{agg}_A^1 \neq \text{agg}_A^2$ fest. Damit sind beide Knoten, A und C, über den Betrug informiert und melden ihre Erkenntnis an E und die folgenden Knoten auf dem Aggregationspfad.
- Knoten D stellt beim Mehrheitsentscheid von agg_A fest, dass $\text{agg}_A^2 \neq \text{agg}_A^0 = \text{agg}_A^1$ gilt. Deshalb ist C korrumpiert. D meldet dies an A und B und hängt seine Schlüssel E_{AD}, E_{BD} und E_{CD} an. Knoten A kann damit seine protokollierten Daten entschlüsseln und stellt $\text{agg}_A^2 \neq \text{agg}_A^0$ fest. Knoten B stellt $\text{agg}_A^2 \neq \text{agg}_A^1$ fest. Damit sind beide Knoten, A und B, über den Betrug informiert und melden ihre Erkenntnis an E und die folgenden Knoten auf dem Aggregationspfad.

Damit ist sichergestellt, dass ein Betrugsversuch allen legitimen Knoten des Aggregationspfades mitgeteilt wird und keine Fehlalarmierungen durch korrumpierte Knoten, wie sich bei ESAWN möglich sind, auftreten.

3.2.1.1 Beispiel mit $\bar{k} = 1$

Um die Kommunikation der Knoten untereinander noch einmal im Zusammenhang eines größeren Beispiels darzustellen, wird auf das ESAWN-2-Beispiel der Abbildung 3.1 mit $\delta = 1$ und $\bar{k} = 1$ zurückgegriffen. Diese Darstellung hat auch bei ESAWN-NR ihre Gültigkeit, da die dargestelltenMehrheitsentscheide bei ESAWN-NR genau so durchgeführt werden.

Zwei Ablaufdiagramme zeigen die Kommunikation der ersten Knoten dieses Netzwerks detailliert. In der ersten Abbildung 3.7 wird die theoretische Kommunikation dargestellt, also die logischen Kommunikationsverbindungen zwischen zwei Knoten. Dies entspricht auch genau den Verbindungen, die bei ESAWN-2 aufgebaut werden. In der zweiten Abbildung 3.8 ist die tatsächliche Kommunikation ersichtlich. Diese unterschiedet sich deutlich, sowohl von der ersten Darstellung, als auch von der Darstellung des ESAWN-2-Protokolls.

Ein Zusammenfassen der Daten, wie es bei ESAWN-2 möglich war, ist bei ESAWN-NR nicht mehr in diesem Maße möglich, da die Berechnung und der Mehrheitsentscheid des Aggregates für Knoten A erst vollständig abgeschlossen werden muss, bevor dies für das Aggregat von Knoten B möglich ist. Die Schritte können nicht mehr parallel, wie bei ESAWN-2, durchgeführt werden, sondern streng sequentiell.

Daher muss für ESAWN-NR mit deutlich höheren Kosten, im Vergleich zu ESAWN-2, gerechnet werden.

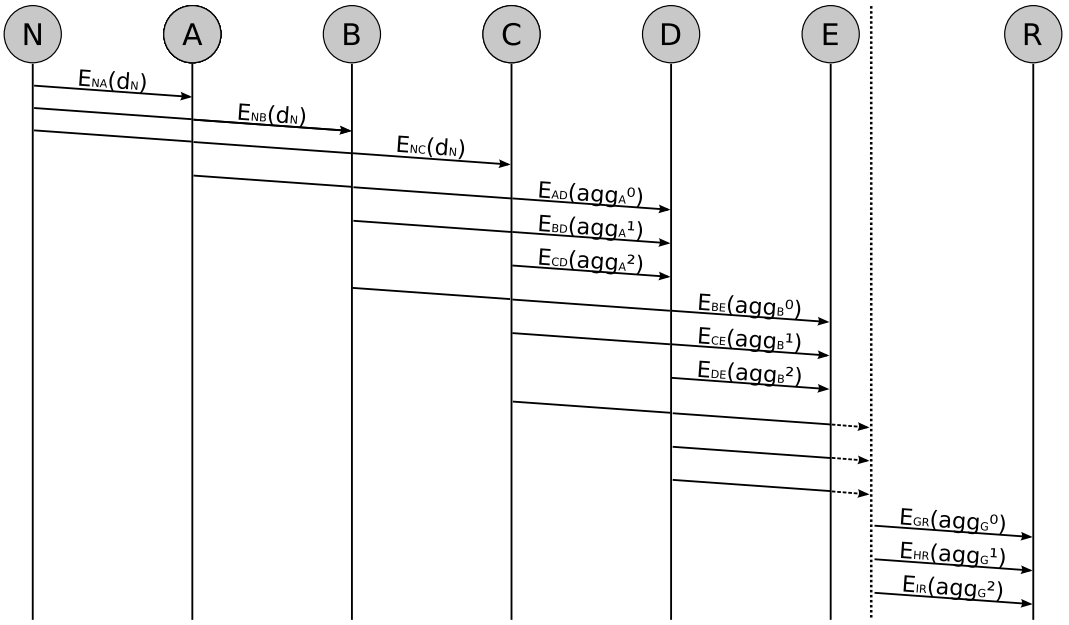


Abbildung 3.7: Visualisierung der theoretischen Kommunikation bei ESAWN-NR

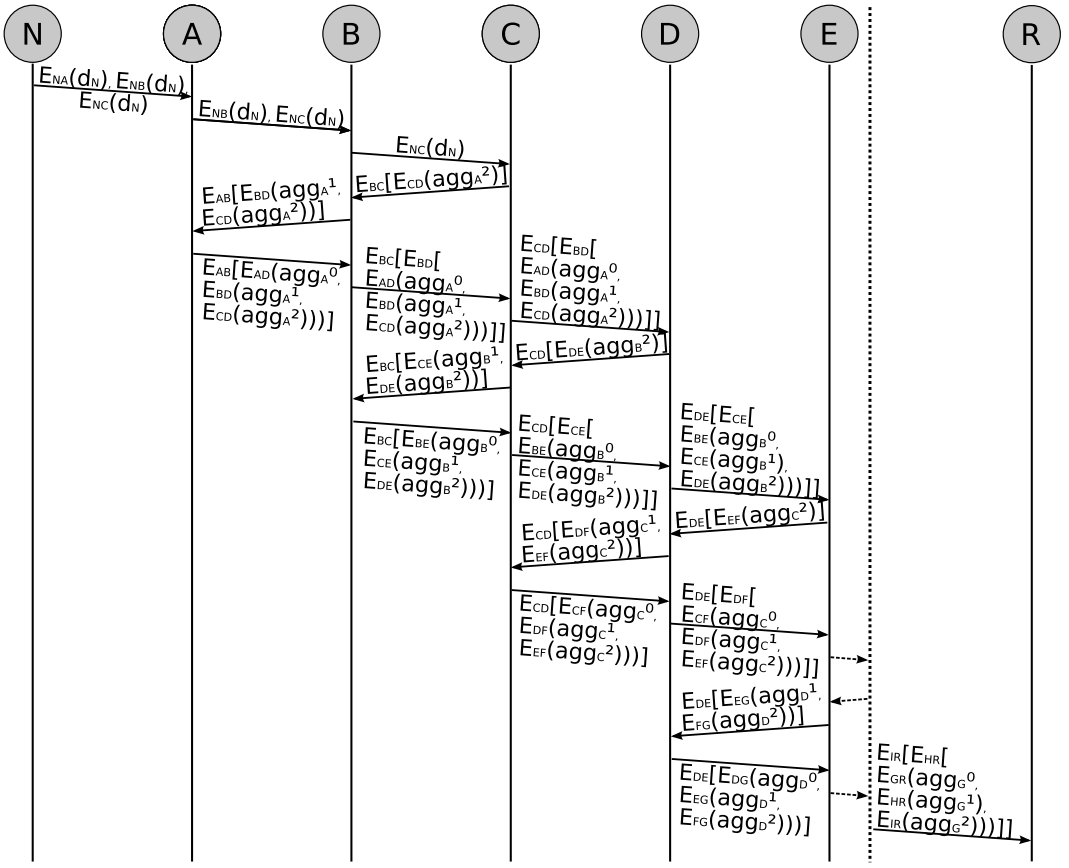


Abbildung 3.8: Visualisierung der tatsächlichen Kommunikation bei ESAWN-NR

3.2.2 Beweis der Korrektheit, Vollständigkeit und Non-Repudiation

Im Folgenden wird bewiesen, dass ESAWN-NR tatsächlich die Leistungsmerkmale von ESAWN-2 mit zusätzlicher Non-Repudiation im Sinne nachfolgender Definition erreicht.

3.2.2.1 Voraussetzungen

Gegeben

- Parameter $\bar{k}$

Definitionen

- Mit der **Schreibweise** v_i wird der i-Vorgänger des Knotens v bezeichnet.
- Unter den **zuständigen Knoten eines Aggregationsschritts** sind der Aggregationsknoten v und seine Vorgänger $v_1, \ldots, v_i, \ldots, v_{2\bar{k}}$ (Zeugen) zu verstehen.
- d_i kennzeichnet das **Datum** welches der Knoten v_i an den Knoten $v_{2\bar{k}+1}$ sendet.
- Jedes von v_i versandte Datum d_i passiert mehrere Knoten und kann dabei gegebenenfalls manipuliert werden, wenn einem dazwischenliegender Knoten der Schlüssel bekannt ist. Dies ist möglich, da einem korrumpierten Knoten auch alle Schlüssel der anderen korrumpierten Knoten zur Verfügung stehen. Der **Inhalt des Datums**, den ein Knoten v_j für das Datum d_i empfängt, wird daher mit d_i^j bezeichnet. Wenn ein Knoten beim Transport des Datums von v_i nach v_j den Inhalt des Datums erfolgreich verändern konnte, gilt $d_i \neq d_i^j$.
- $E_{a,b}(d_i)$ kennzeichnet ein **verschlüsseltes Datum** d_i, das durch einen symmetrischen Schlüssel verschlüsselt ist, den zwei Knoten v_a und v_b miteinander teilen.
- Knoten **protokollieren** Daten, wenn sie diese zu ihrem Elternknoten weiterleiten. Dabei ist es egal, ob das Datum im Klartext oder verschlüsselter Form erfasst wird. Das Datenpaket an sich wird gespeichert.
- **Non-Repudiation** bei ESAWN-NR bedeutet, dass ein Sensorknoten v_i das Versenden eines Datum d_i an einen legitimen Entscheider $v_{2\bar{k}+1}$ gegenüber allen Zuständigen eines Aggregationsschritts nicht abstreiten kann. Durch offenlegen der Kommunikationsschlüssel durch den Entscheider, kann jedem zuständigen Knoten der Versand eines bestimmten Datum nachgewiesen werden.

3.2.2.2 Beweis der Vollständigkeit und Korrektheit

Der grundlegende Ablauf bei ESAWN-NR unterscheidet sich nicht von dem bei ESAWN-2. Es werden dieselben Daten von denselben Sendern an dieselben Empfänger gesendet und dadurch auch dieselben Mehrheitsentscheide durchgeführt. Unterschiedlich ist der Weg, den die Pakete dabei nehmen und die Art und Weise wie sie dabei verschlüsselt werden.

Damit lässt sich der Beweis zu ESAWN-2 auf ESAWN-NR übertragen. Die Modifikation der Kommunikationsreihenfolge und die zusätzlichen Verschlüsselungen haben keinen Einfluss drauf. ESAWN-NR besitzt also hinsichtlich Vollständigkeit und Korrektheit dieselben Eigenschaften wie ESAWN-2.

3.2.2.3 Beweis der Non-Repudiation

Um zu beweisen, dass ESAWN-NR Non-Repudiation im Sinne der obigen Definition bietet, wird schrittweise vorgegangen. Zuerst wird gezeigt, dass alle zuständigen Knoten des Aggregationsschrittes für alle dabei sendenden Knoten das gleiche Datum d_i protokollieren, wenn auch unterschiedlich verschlüsselt. Ist dies der Fall, kann der Entscheiderknoten in diesem Aggregationsschritt durch Offenlegen seiner Kommunikationsschlüssel den anderen Knoten beweisen, dass ein Knoten v_i wirklich d_i gesendet hat. Sind die $\bar{k}+1$ oder mehr legitimen Knoten dieses Aggregationsschritts über den Betrugversuch informiert, teilen sie ihre Erkenntnis an die nächsten $2\bar{k}+1$ Knoten auf dem Aggregationspfad Richtung Senke mit. Die Knoten akzeptieren solch eine Meldung, wenn sie diese von hinreichend vielen, das heißt $> \bar{k}$ aus $2\bar{k}+1$, hintereinanderliegenden Knoten erreicht.

Es wird ein beliebiger Aggregationsschritt im Aggregationsbaum B betrachtet. Bei diesem Aggregationsschritt ist die Menge der Knoten V zuständig. V enthält den Aggregationsknoten v_0 und seine Zeugen $v_1, \ldots, v_{2\bar{k}}$.

V lässt sich in zwei Teilmengen zerlegen, V_K und V_L. Dabei gilt:

1. $\forall v \in V_L$ gilt: v ist legitim
2. $\forall v \in V_K$ gilt: v ist korrumpiert
3. $|V_K| \leq \bar{k}$, dies folgt aus der Annahme $A_{\bar{k}}$

Die Punkte 1 und 2 klassifizieren die beiden Mengen V_L und V_K als Menge legitimer und korrumpierter Knoten. Punkt 3 fordert zusätzlich, dass maximal $\bar{k}$ Knoten korrumpiert sind und berücksichtigt somit, dass ESAWN-NR nur bis zu dieser Zahl an korrumpierten Knoten fehlerfrei arbeitet.

Auch vom Entscheiderknoten, $v_{2\bar{k}+1}$, wird angenommen, dass dieser legitim ist. Dies taucht schon in der Definition der von ESAWN-NR realisierten Non-Repudiation auf. Dies mag auf den ersten Blick eine unangemessen hohe Forderung darstellen. Tatsache ist jedoch, dass ein korrumpierter Entscheiderknoten unabhängig von seinen Eingabedaten handelt. Es ist nicht von Vorteil, wenn dem korrumpierten Absenderknoten nachgewiesen werden kann, was er an den ebenfalls korrumpierten Entscheider-Knoten gesendet hat.

Insbesondere kann es in keinem Fall dazu kommen, dass einem legitimen Knoten dadurch ein fehlerhaftes Aggregat „untergeschoben“ wird. Sobald der korrumpierte Absenderknoten sein fehlerhaftes Aggregat – im Rahmen eines anderen Aggregationsschritts – an einen legitimen Entscheider-Knoten sendet, wird dieser das entdecken und eine entsprechende Alarmierung an die anderen beteiligten Knoten versenden. Dieser Entscheider muss, dem Protokoll gemäß, den oder die korrumpierten Knoten, zusammen mit den benötigten Schlüsseln, melden.

Behauptung

Für alle Teilmengen $V_K \subseteq V$ und jeden Knoten $v_i \in V$ gilt:

1. Alle Knoten $\in V$ protokollieren für das Datum d_i den gleichen Inhalt.
   $(\forall j,k \in \{0,1,\ldots,2\bar{k}+1\} : d_i^j = d_i^k)$
2. Wenn v_i legitim ist, protokollieren alle Knoten das korrekte Datum.
   $(v_i \in V_L \Rightarrow \forall j \in \{0,1,\ldots,2\bar{k}+1\} : d_i = d_i^j)$
3. v_i kann gegenüber allen Knoten $\in V_L$ nachgewiesen werden, dass er d_i versandt hat („Non-Repudiation “).

Beweis der Behauptung 1

- Fall 1 – $v_j \in V_L$:
  v_j verschlüsselt sein Datum d_j, gemäß Protokollvorschrift, an $v_{2\bar{k}+1}$: $E_{j,2\bar{k}+1}(d_j)$. Dieses Datenpaket können nur Absender und Empfänger verändern, daher gilt: $d_j^0 = d_j^1 = \ldots = d_j^{2\bar{k}+1} \Rightarrow$ Behauptung 1.
- Fall 2 – $v_j \in V_K$:
  v_j verschlüsselt sein Datum d_j, gemäß Protokollvorschrift, an $v_{2\bar{k}+1}$: $E_{j,2\bar{k}+1}(d_j)$. Dieses Datenpaket können der Absender, der Empfänger und auch alle anderen korrumpierten Knoten verändern. Annahme: Es gibt einen Knoten $v_a \in V_L$ für den gilt: $d_j^a \neq d_j^{2\bar{k}+1}$.
  Dann muss es einen Knoten $v_b \in V_K$ geben, mit $a < b < 2\bar{k}+1$, der in dem von a versandten Datenpaket $E_{a,2\bar{k}+1}(E_{\ldots}(d_j)\ldots)$ das Datum ändern kann. Da das Datum unter anderem durch die Verschlüsselung $E_{a,2\bar{k}+1}(\ldots)$ geschützt ist und die beteiligten Knoten a und $2\bar{k}+1$ beide legitim sind, ist dies nicht möglich. Widerspruch.

Beweis der Behauptung 2

v_j ist legitim und verschlüsselt sein Datum d_j an $v_{2\bar{k}+1}$: $E_{j,2\bar{k}+1}(d_j)$. Dieses Datenpaket können nur Absender und Empfänger verändern, daher gilt: $d_j = d_j^0 = d_j^1 = \ldots = d_j^{2\bar{k}+1} \Rightarrow$ Behauptung 2.

Beweis der Behauptung 3

Alle Knoten $v_j \in V_L$ protokollieren die gleichen Daten in einem Aggregationsschritt. Der Entscheider löst bei fehlender Übereinstimmung der Aggregate beim Mehrheitsentscheid Alarm aus und informiert alle Knoten $\in V_L$. Dazu legt er alle notwendigen Kommunikationsschlüssel mit den zuständigen Knoten offen und ermöglicht so allen Knoten $\in V_L$ die Überprüfung der Alarmmeldung. Jeder Knoten kann dadurch feststellen, ob das von v_i versendete Datum korrekt war $\Rightarrow$ Behauptung 3.

3.2.3 Prüfwahrscheinlichkeit p für Aggregationsknoten

Wie bei ESAWN-2 ist auch bei ESAWN-NR die Einführung einer Prüfwahrscheinlichkeit ein effektiver Schritt, die Energiekosten durch weniger zu übertragende Daten sowie Aggregations- und Kryptographieoperationen zu senken. p findet bei ESAWN-NR auf die gleiche Weise Anwendung, wie bei ESAWN-2. Zur Referenz sei auf Abschnitt 3.1.5 verwiesen.

3.3 Zusammenfassung

ESAWN-2 bietet funktionale Vorteile gegenüber ESAWN. Insbesondere verhindert das neue Protokoll Fehlalarme und abgebrochene Messvorgänge. Dies wird sich jedoch in einem höheren Energie- und Speicherverbrauch niederschlagen, da deutlich mehr Aggregate berechnet und kommuniziert werden müssen, als dies bei ESAWN der Fall ist.

ESAWN-NR erweitert das Funktionsspektrum nochmals und ermöglicht, mittels Non-Repudiation, korrumpierte Knoten gezielt zu identifizieren und dies im Netzwerk zu kommunizieren. Durch die Veränderung der Kommunikationsabfolge geht hierbei jedoch die Möglichkeit verloren, Datenpakete mehrerer Aggregationsschritte zusammenzufassen. Außerdem müssen die Knoten für die Non-Repudiation mehr Daten speichern. Deshalb muss für ESAWN-NR mit nochmals höherem Energie- und Speicherverbrauch gerechnet werden, den es im übernächsten Kapitel zu evaluieren gilt.

Wie bereits bei ESAWN bieten ESAWN-2 und ESAWN-NR mit dem Systemparameter $\bar{k}$ die Möglichkeit an, den Tradeoff zwischen Authentizität und Energieverbrauch selbst anwendungsspezifisch zu regulieren.

Beide Verfahren haben mit dem Systemparameter p außerdem die gleichen Möglichkeit wie ESAWN erhalten durch eine Reduzierung der Prüfwahrscheinlichkeit die Energiekosten zu senken. Ob sich dies bewährt, wird sich im Rahmen der Evaluation herausstellen.

4. Implementierung

Ziel des Kapitels: Um eine sinnvolle Bewertung und Analyse der ESAWN-Protokolle zu ermöglichen, wurden diese auf verschiedene Weisen implementiert. Neben dem Entwurf der Protokolle in Pseudocode erfolgte eine Implementierung in GloMoSim [42], einer skalierbaren Simulationsumgebung für drahtlose Netzwerke. Aufgrund der Komplexität und dem damit verbundenen Zeitaufwand der GloMoSim-Simulationsläufe wurde zusätzlich ein zweites, selbst erstelltes Programm eingesetzt. Zuletzt wurde eine reale Sensorknotenplattform betrachtet. Die Wahl fiel auf MICAz-Sensorknoten und das Betriebssystem TinyOS. Die Implementierung der ESAWN-Protokolle und die darauf folgenden Tests zeigen die praktische Realisierbarkeit und ermöglichen eine Demonstration mit echten Sensorknoten.

Dieses Kapitel stellt die verschiedenen Implementierungen vor.

Aufbau des Kapitels: Zuerst werden die Protokolle in Pseudocode vorgestellt (Abschnitt 4.1). In Abschnitt 4.2 wird die GloMoSim-Implementierung vorgestellt. Anschließend wir in Abschnitt 4.3 auf das selbst erstellte Programm SecuSim eingegangen. Den Abschluss des Kapitels bildet die Vorstellung der Implementierung in TinyOS in Abschnitt 4.4.

4.1 Pseudocode

Vor der eigentlichen Präsentation der Protokolle in Pseudocode werden einige Vorgaben erläutert, die das Verständnis des Pseudocodes unterstützen sollen. Anschließend wird zuerst auf ESAWN-2 und dann auf ESAWN-NR im Detail eingegangen.

4.1.1 Vorgaben

Zuerst ein kurzer Überblick über die im Pseudocode verwendeten Notationen, Datentypen, globalen Variablen und vordefinierten Methoden.

Notationen

- $a \leftarrow b$ – Der Variable a wird der Wert der Variablen b zugewiesen.

Datentypen

- `bool` – Ein Wahrheitswert, kann *true* oder *false* annehmen
- `int` – Umfasst alle Elemente aus $\mathbb{Z}$.
- `float` – Umfasst alle Elemente aus $\mathbb{R}$.
- `NodeId` – Speichert eine Knotenadresse.
- `DataElement` – Bestandteil eines Datenpakets, enthält die Attribute `int value` (speichert ein Datum) und `NodeId id` (speichert eine Knotenadresse).
- `List<t>` – Eine Liste mit Elementen vom Typ t. Der Liste können mit `add(t element)` neue Elementen angehängt werden. Die Größe der Liste liefert `size()` und `clear()` löscht die Liste. `containsElement(Kriterium)` überprüft die Liste auf Enthaltensein eines bestimmten Elements. `getElements(Kriterium)` liefert alle Elemente der Liste die einem bestimmten Kriterium entsprechen zurück.

Globale Variablen

Folgende Variablen sind aus dem gesamten Code erreichbar:

- `NodeId me` – Die eigene Adresse des Sensorknoten.
- `List<DataElement> dataStore` – Datenspeicher für verifizierte Daten.
- `List<DataElement> dataStoreME` – Datenspeicher für die Daten, die noch durch Mehrheitsentscheid verifiziert werden müssen.
- `List<DataElement> queue` – Warteschlange zu versendender Daten.
- `List<DataElement> corruptedNodes` – Enthält Informationen über korrumpierte Knoten.
- `int inCount` – Enthält die Zahl der eingegangenen Datenelemente.
- `int step` – Enthält den momentan durchgeführten Teilschritt des Protokolls (nur ESAWN-NR). Vor Protokollstart gilt step=$2\bar{k} + 2$.
- `bool doLogging` – Speichert, ob der Knoten die ankommenden Daten protokollieren muss (nur ESAWN-NR).

Vordefinierte Methoden

Die genaue Implementierung einiger Funktionen auf die im Pseudocode Bezug genommen wird, ist für die Beschreibung der Funktionsweise von ESAWN-2 und ESAWN-NR nicht im Detail relevant. Die betreffenden Funktionen sind im nachstehender Auflistung mit einer kurzen Funktionsbeschreibung aufgeführt.

- `NodeId getParent(NodeId n)` – Liefert den Vaterknoten von n im Aggregationsbaum zurück.

- `List<NodeId> getChildList(NodeId n)` – Liefert eine Liste mit den Knoten-IDs der Kindknoten von n im Aggregationsbaum.
- `NodeId predecessor(NodeId id, int s)` – Liefert den s-Vorgänger des Knoten *id* zurück.
- `int getHeight(NodeId id)` – Liefert die Höhe des Knoten *id* im Aggregationsbaum zurück.
- `bool isLeaf(NodeId n)` – Gibt *true* zurück, wenn n ein Blattknoten ist, sonst *false*.
- `decrypt(NodeId n, List<DataElement> list)` – Entschlüsselt die Datenelemente in *list* mit dem Schlüssel, den sich der aktuelle Knoten mit Knoten n teilt.
- `int encrypt(NodeId n, List<DataElement> list)` – Verschlüsselt die Datenelemente in *list* mit dem Schlüssel, den sich der aktuelle Knoten mit Knoten n teilt.
- `bool checkAgg(NodeId n)` – Gibt *true* zurück, wenn Knoten n überprüft werden muss, sonst *false*.
- `int getMajority(List<int> values)` – Liefert den Wert zurück, der mehrheitlich in der Liste *values* vorkommt.
- `sendDataToParent(List<DataElement> queue)` – Sendet alle Datenelemente in *queue* an den Vaterknoten des aktuellen Knotens. Leert *queue*.
- `sendDataToChildren(List<DataElement> queue)` – Sendet alle Datenelemente in *queue* an die jeweiligen Kindknoten des aktuellen Knotens. Leert *queue*.
- `int doMeasure()` – Liefert einen Messwert des Knotensensors.
- `int calcAggregate(NodeId n, List<DataElement> dataStore)` – Aggregiert alle in *dataStore* für Knoten n enthaltenen Daten und liefert das Ergebnis zurück.
- `saveToLog(DataElement dat)` – Speichert das verschlüsselte Datenelement für später eventuell anfallende Verifizierungsmaßnahmen (Non-Repudiation bei ESAWN-NR).
- `sendAlarmAndProof(NodeId id, NodeId corruptedId, NodeId aggId)` – Sendet an Knoten *id* die Information, dass *corruptedId* versucht hat, den Aggregationsschritt *aggId* zu manipulieren und hängt die notwendigen Kommunikationsschlüssel an die Nachricht an.
- `sendAlarmNoProof(NodeId id, NodeId corruptedId)` – Sendet Knoten *id* die Information, dass *corruptedId* korrumpiert ist.
- `reportToBase(int value)` – Meldet der Basisstation *value* als Gesamtaggregat der Senke. Diese Funktion kann nur von der Senke genutzt werden.

4.1.2 ESAWN-2

Im Folgenden ein detaillierter Einblick in die Methoden und ihre Funktionsweise bei der Verwendung von ESAWN-2.

Prozedur `start`

```
   /* Deklarationen                                                    */
1  DataElement dat
2  NodeId id
3  int value
   /* Messdatum erfassen                                               */
4  value ← doMeasure()
5  id ← getParent(me)
   /* Datum an die 2k̄ + 1 nächsten Vorgänger versenden                 */
6  for i ← 1 to 2k̄ + 1 do
7      dat.value ← value
8      encrypt(id, dat)
9      dat.id ← me
10     queue.add(dat)
11     if id = 0 then
12         exit for loop
13     id ← getParent(id)
14 sendDataToParent(queue)
```

start()

Diese Methode wird auf jedem Blattknoten aufgerufen, um einen Messvorgang und den Ablauf der Aggregation zu starten. Nach der sensorischen Erfassung des Datums (Zeile 3) wird das Datum für alle $2\bar{k} + 1$ Empfänger verschlüsselt (Zeile 6). Als Absender-ID wird die eigene Knoten-ID hinzugefügt (Zeile 7). Jedes Datum wird an die Warteschlange *queue* angehängt (Zeile 8). Sollten nicht genügend Knoten bis zur Senke vorhanden sein, werden weniger Daten verschickt (Zeile 9-10). Abschliessend werden alle Daten an den direkten Vorgänger gesendet (Zeile 12).

Diese Methode ist bei ESAWN-2 und ESAWN-NR gleich.

Esawn2.receiveData()

Diese Methode wird aufgerufen, wenn ein Knoten ein ESAWN Datenpaket *dataPacket* erhält. Die eingehenden Daten werden dabei zuerst durch die Methode *storeOrForward* (Zeile 5) verarbeitet, die alle Daten in drei Gruppen speichert: weiterleiten (*queue*), durch Mehrheitsentscheid verifizieren (*dataStoreME*) und direkt speichern (*dataStore*). Die für den Knoten selbst bestimmten Daten werden dabei sofort dechiffriert. Anschliessend werden zwei Listen, *nList* und *wList*, aufgebaut, die alle Knoten enthalten von denen Daten erwartet werden oder für die eine Zeugenaggregation durchgeführt werden muss (Zeile 6-7).

In Zeile 8 wird geprüft, ob bereits alle Daten von den Kindknoten empfangen wurden. Falls dies nocht nicht der Fall ist, ist die Methode hier zu Ende und wird durch ein nachfolgend eingehendes Datenpaket wieder aufgerufen.

Prozedur `Esawn2.receiveData(`*dataPacket*`)`
Eingabe : List<DataElement> dataPacket

```
   /* Deklarationen                                                    */
1  DataElement dat
2  NodeId id
3  List<NodeId> nList
4  List<NodeId> wList
   /* eingehende Daten in dataStore oder queue speichern               */
5  storeOrForward(dataPacket)
6  getNodes(nList, me, 0)
7  witnessList(wList, me, 0)
   /* Prüfe, ob alle notwendigen Daten empfangen wurden                */
8  if dataStore.size() = nList.size() then
       /* Alle notwendigen Aggregationen durchführen                   */
9      forall id in wList do
10         dat.id ← id
11         dat.value ← calcAggregate(id, dataStore)
12         encrypt(predecessor(id, 2k̄ + 1), dat)
13         queue.add(dat)
14     if me > 0 then
           /* Alle Daten an den Vaterknoten versenden                  */
15         sendDataToParent(queue)
16     else
17         reportToBase(calcAggregate(0, dataStore))
```

Andernfalls kann der Knoten jetzt alle ihm zugewiesenen Aufgaben erledigen. Dazu wird für jede in *wList* enhaltene Knoten-ID eine Zeugenaggregation durchgeführt (Zeile 11) und das Aggregat für den Entscheiderknoten verschlüsselt (Zeile 12). Alle Daten werden der Warteschlange *queue* hinzugefügt (Zeile 13).

Handelt es sich bei dem aktuellen Knoten nicht um die Senke, so werden die Daten der Warteschlange abschliessend an den direkten Vorgänger gesendet (Zeile 15). Andernfalls wird das Ergebnis der Basisstation gemeldet (Zeile 17).

storeOrForward()

Ein Großteil der Aufgaben beim Empfangen der Datenpakete läuft in *storeOrForward()* ab. Diese Methode wird von *Esawn2.receive()* aufgerufen und sorgt dafür, dass die empfangenen Daten richtig zugeordnet und entschlüsselt werden. Diese Methode ist bei ESAWN-2 und ESAWN-NR gleich. Der einzige Unterschied liegt in Zeile 16, da hier eingehende Daten von ESAWN-2 und ESAWN-NR unterschiedlich entschlüsselt werden. Diese Aktion ist deshalb in die Methoden *Esawn2.decryptData()* und *EsawnNR.decryptData()* ausgelagert.

storeOrForward() verarbeitet der Reihe nach jedes Datenelement *dat* des Datenpakets *dataPacket* (Zeile 3). Dabei sind zwei Fälle möglich.

- Die Daten stammen von einem Blattknoten. Liegen dem aktuellen Knoten noch keine Daten von diesem Blattknoten vor, werden die Daten direkt entschlüsselt und in dataStore gespeichert (Zeile 8-9). Andernfalls werden die Daten zur Weiterleitung in *queue* gespeichert (Zeile 6).
- Die Daten stammen nicht von einem Blattknoten. Ist der aktuelle Knoten nicht der Entscheiderknoten für die Absender-ID, werden die Daten zur Weiterleitung in *queue* gespeichert (Zeile 24). Andernfalls wird durch einen Aufruf von *checkAgg()* festgestellt, ob das eingehende Datum überprüft werden muss. Ist dies der Fall, werden die Daten für einen Mehrheitsentscheid vorgemerkt (Zeile 13). Andernfalls wird wie bei einem Blattknoten verfahren (Zeile 21-22).

Immer wenn Daten für einen Mehrheitsentscheid vorgemerkt werden, wird geprüft, ob bereits alle Daten für die Durchführung des Mehrheitsentscheides vorhanden sind (Zeile 14-15). Falls dies der Fall ist, werden die betreffenden Daten entschlüsselt (Zeile 16), der mehrheitlich vorkommende Wert bestimmt und im Datenspeicher *dataStore* abgelegt (Zeile 17).

Esawn2.getNodes()

Esawn2.receive() stellt mit dieser Methode fest, ob bereits alle Daten eingetroffen sind. *Esawn2.getNodes()* erstellt dazu eine Liste mit den Knoten, von denen Daten erwartet werden. Diese Liste enthält sowohl alle relevanten Blattknoten (bis zu $2\bar{k}+1$ Schritte entfernt) und alle Knoten für die der aktuelle Knoten Entscheiderknoten ist.

Die Methode ist rekursiv definiert und führt eine Tiefensuche über den betroffenen Teilbaum mit dem aktuellen Knoten als Wurzelknoten durch. Die relevanten Knoten werden dabei in Zeile 8 (Entscheider) und in Zeile 11 (Blattknoten) identifiziert und der Ergebnisliste hinzugefügt.

Prozedur `storeOrForward(`*dataPacket*`)`

Eingabe : List<DataElement> dataPacket

```
   /* Deklarationen                                                    */
1  DataElement dat
2  List<DataElement> list
   /* Verarbeite jedes Element des Datenpakets                         */
3  forall dat in dataPacket do
4      if isLeaf(dat.id) then
5          if dataStore.containsElement(id = dat.id) then
               /* Datum liegt schon vor, weiterleiten                  */
6              queue.add(dat)
7          else
               /* dechiffrieren und speichern                          */
8              decrypt(dat.id, dat)
9              dataStore.add(dat)
10     else
11         if predecessor(dat.id, 2k̄ + 1) = me then
               /* Datum sofort oder für späteren Mehrheitsentscheid
                  speichern                                            */
12             if checkAgg(dat.id) then
13                 dataStoreME.add(dat)
14                 list ← dataStoreME.getElements(id = dat.id)
15                 if list.size() = 2k̄ + 1 then
16                     decryptData(list)
                       /* Mehrheitsentscheid durchführen               */
17                     dataStore.add(getMajority(list))
18                     dataStoreME.clearElements(id = dat.id)
                       /* Nur bei ESAWN-NR: ggf. Alarmierung           */
19                     initAlarm(list)
20             else
21                 decrypt(dat.id, dat)
22                 dataStore.add(dat)
23         else
               /* Rest weiterleiten                                    */
24             queue.add(dat)
25 inCount ← inCount + dataPacket.size();
```

Prozedur `Esawn2.getNodes(`*dList, child, h*`)`

Eingabe : List<NodeId> nList, NodeId child, int h
Ausgabe : List<NodeId> nList

```
   /* Lokale Deklarationen                                          */
1  List<NodeId> childList
   /* Rekursives Durchlaufen des Aggregationsbaums                  */
2  if h = 0 then
3      dList.clear()
4  childList ← getChildList(child)
5  forall child in childList do
6      if isLeaf(child) = false then
7          if h = 2k̄ + 1 then
8              nList.add(child)
9          else
10             getNodes(nList, child, h+1)
11     else  nList.add(child)
```

Prozedur `Esawn2.witnessList(`*wList, child, h*`)`

Eingabe : List<NodeId> wList, NodeId child, int h
Ausgabe : List<NodeId> wList

```
   /* Lokale Deklarationen                                          */
1  List<NodeId> childList
   /* Rekursives Durchlaufen des Aggregationsbaums                  */
2  if h = 0 then
3      wList.clear()
4  childList ← getChildList(child)
5  forall child in childList do
6      if h ≤ 2k̄ then
7          if isLeaf(child) = false and checkAgg(child) then wList.add(child)
8          witnessList(child, h+1)
9  if h = 0 and predecessor(me, 2k̄) > 0 then
10     wList.add(me)
```

Esawn2.witnessList()

Diese Methode bestimmt alle Knoten, für die der aktuelle Knoten Zeugenaggregationen durchführen muss. Genau wie in *Esawn2.getNodes()* wird rekursiv eine Tiefensuche über den Aggregationsbaum durchgeführt. Alle relevanten Knoten werden der Ergebnisliste hinzugefügt (Zeile 7). Dies ist unter Umständen auch der Knoten selbst (Zeile 10).

Prozedur `Esawn2.decryptData(`*list*`)`

Eingabe : List<DataElement> list
Ausgabe : List<DataElement> list

```
   /* Deklarationen                                          */
1  DataElement dat
2  NodeId id
3  NodeId decryptId
4  decryptId ← id
5  forall dat in list do
6  |  decrypt(decryptId, dat)
7  |  decryptId ← getParent(decryptId)
```

Esawn2.decryptData()

Esawn2.decryptData() entschlüsselt eine Liste von Werten, die von *storeOrForward()* für einen Mehrheitsentscheid benötigt werden. Da alle in der Liste enthaltenen Werte die gleiche Absender-ID enthalten, schliesst die Methode aufgrund der Reihenfolge auf den für die Entschlüsselung zu verwendenden Schlüssel. Das erste Element stammt dabei vom Aggregator selbst, die folgenden Elemente von seinen Zeugen.

4.1.3 ESAWN-NR

Im Pseudocode zu ESAWN-NR werden zwei zusätzliche globale Variablen benötigt: *step* und *doLogging* (siehe Abschnitt 4.1.1). Diese werden benötigt um den Zustand in dem sich das Protokoll befindet zu speichern um damit den sequentiellen Ablauf des Protokolls zu steuern.

step enthält dabei Werte zwischen $2\bar{k}+1$ und 0. *step*=$2\bar{k}+1$ entspricht dabei der Phase des Weiterleitens der Messwerte von Blattknoten. *step*=$2\bar{k}$,...,1 stellt die Phasen der Aggregation als Zeuge der Knoten mit Abstand $2\bar{k}+1$,...,1 dar. Zuletzt *step*=0, hier bildet der Knoten selbst sein Aggregat und schliesst den Messvorgang ab.

doLogging enthält *true*, wenn der Knoten die empfangenen Pakete für Zwecke der „Non-Repudiation“ speichern muss. Dies ist immer dann der Fall, wenn Daten Richtung Wurzel transportiert werden.

start()

Wie bei ESAWN-2 wird der Aggregationsvorgang durch die sensorische Erfassung der Messdaten in den Blattknoten begonnen. Dabei kommt exakt der gleiche Pseudocode zum Einsatz wie bei ESAWN-2.

Prozedur `EsawnNR.receiveData(`*dataPacket*`)`

Eingabe : List<DataElement> dataPacket

```
   /* Lokale Deklarationen                                                    */
1  List<NodeId> nList
2  DataElement dat
3  int count
   /* eingehende Daten in dataStore oder queue speichern                     */
4  storeOrForward(dataPacket)
5  getNodes(nList, me, step, 0)
6  count ← 0
7  if step = 2k̄ + 1 then
8      forall id in nList do
9          if isLeaf(id) then
10             int m ← 2k̄ + 1
11             if getHeight(id) < m then m ← getHeight(id)
12             count ← count + m + 1 - getHeight(id) +getHeight(me)
13         else
14             count ← count + 1
15             if checkAgg(id) = true then  count ← count + 2k̄
16 else
17     if doLogging = true then
18         forall id in nList do
19             count ← count + 1
20             if checkAgg(id) = true then count ← count + 2k̄
21     else
22         forall id in nList do
23             if checkAgg(id) = true then count ← count + 2k̄ - step
24 if count = inCount then  processStep()
```

EsawnNR.receiveData()

EsawnNR.receiveData() wird aufgerufen, wenn ein Knoten ein ESAWN Datenpaket *dataPacket* erhält. Die eingehenden Daten werden dabei zuerst durch die Methode *storeOrForward* (Zeile 3) verarbeitet, die alle Daten in drei Gruppen speichert: weiterleiten (*queue*), durch Mehrheitsentscheid verifizieren (*dataStoreME*) und direkt speichern (*dataStore*). Die für den Knoten selbst bestimmten Daten werden dabei sofort dechiffriert.

Anschliessend wird geprüft, ob, abhängig von *step* und *doLogging*, alle notwendigen Daten für die Weiterverarbeitung eingetroffen sind (Zeile 7-23). Dabei wird unterschieden, ob der Knoten sich im Zustand $step{=}2\bar{k}+1$ befindet (Zeile 8-15) oder in einem darauf folgenden Zustand (Zeile 17-23). Im ersten Fall resultiert die Zahl der erwarteten Datenelemente aus dem Abstand zu den jeweiligen Blattknoten (Zeile 12). Im zweiten Fall hängt sie von der Richtung des Transports (*doLogging*) ab und der Tatsache, ob der betrachtete Aggregationsknoten überhaupt geprüft wird (Zeile 20, 23).

Die erwartete Zahl an Datenelementen wird in *count* gespeichert und mit dem tatsächlichen Wert in *inCount* verglichen (Zeile 24). Sind alle Datenelemente eingetroffen, kann die Verarbeitung in *processStep()* erfolgen.

storeOrForward()

Das in Zeile 3 von *EsawnNR.receiveData()* aufgerufene *storeOrForward()* ist identisch mit dem von ESAWN-2. Einzig die in *storeOrForward()* genutzte Methode *decryptData()* ist bei ESAWN-NR abgeändert und wird im nächsten Absatz besprochen.

Prozedur `EsawnNR.decryptData(`*list*`)`

Eingabe : List<DataElement> list
Ausgabe : List<DataElement> list

```
   /* Deklarationen                                          */
 1 DataElement dat
 2 NodeId id
 3 NodeId decryptId
 4 int i
 5 for i ← list.size() − 1 to 0 do
 6 |   decryptId ← predecessor(dat.id, i)
 7 |   decrypt(decryptId, list)
 8 decryptId ← id
 9 forall dat in list do
10 |   decrypt(decryptId, dat)
11 |   decryptId ← getParent(decryptId)
```

EsawnNR.decryptData()

EsawnNR.decryptData() entschlüsselt eine Liste von Werten, die von *storeOrForward()* für einen Mehrheitsentscheid benötigt werden. Bevor die Daten analog zu

ESAWN-2 entschlüsselt werden können, müssen die Verschlüsselungsebenen entschlüsselt werden, die durch den Weitertransport der Aggregate zum Entscheiderknoten dazugekommen sind (Zeile 5-8). Wie bei ESAWN-2 schliesst die Methode aufgrund der Reihenfolge auf den für die Entschlüsselung zu verwendenden Schlüssel (Zeile 9-12).

EsawnNR.processStep()

processStep sorgt dafür, dass abhängig von dem aktuellen Teilschritt *step*, die richtigen Zeugenaggregationen durchgeführt und an die korrekten Knoten versendet werden.

Für *step*=$2k+1$ bedeutet dies, die Messdaten der Blattknoten weiterzuleiten (Zeile 6). Anschliessend wird *step* angepasst, da dieser Teilschritt damit abgeschlossen ist (Zeile 7-11).

Enthält *step* dagegen einen niedrigeren Wert müssen zuerst alle notwendigen Zeugenaggregate berechnet werden (Zeile 21-28) und zu den Aggregatoren geschickt werden. Falls dies dagegen schon geschehen ist, das heisst *doLogging* = *true* gilt, werden die Daten verschlüsselt und zum Entscheiderknoten weitertransportiert (Zeile 30-35). Einen Sonderfall bildet *step=0*, da hier das eigene Aggregat berechnet werden muss (Zeile 13-19).

Abschliessend wird *step* wieder angepasst, da der Teilschritt mit dieser Aktion abgeschlossen ist (Zeile 34-35).

EsawnNR.getNodes()

EsawnNR.getNodes() liefert eine Liste mit relevanten Knoten für den aktuellen Teilschritt der Aggregation. Sie enthält entweder die Blattknoten für die der aktuelle Knoten Daten weiterleiten muss (Zeile 7) oder die Aggregationsknoten mit Abstand *step* für die der aktuelle Knoten Zeuge ist (Zeile 9).

EsawnNR.initAlarm()

EsawnNR.initAlarm() prüft die Werte in *list* auf Einheitlichkeit. Falls es Werte gibt, die von dem mehrheitlich vorkommenden Wert abweichen (Zeile 6), wird durch *sendProof()* eine Schlüsselaufdeckung durchgeführt, in dem die Kommunikationsschlüssel an die zuständigen Knoten gesendet werden (Zeile 7-11). Diese überprüfen anhand der mit *saveToLog()* protokollierten Daten, ob die Alarmmeldung legitim ist. Falls dies der Fall ist, wird auf dem Empfängerknoten *EsawnNR.sendAlarm(corruptedId)* aufgerufen, um die Information auch an alle anderen Knoten auf dem Aggregationspfad weiterzugeben.

EsawNR.sendAlarm()

EsawnNR.sendAlarm() verbreitet die Information über einen korrumpierte Knoten auf dem Aggregationspfad in Richtung Senke weiter. Dazu wird die Information an alle Vorgänger gesendet, mit denen der aktuelle Knoten kommunizieren kann (maximal $2\bar{k}+1$ Stück). Beim Empfänger einer solchen Nachricht wird *EsawnNR.receiveAlarm()* aufgerufen.

Prozedur `EsawnNR.processStep(`*nList*`)`

Eingabe : List<NodeId> nList

```
/* Lokale Deklarationen                                              */
1  int count
2  DataElement dat
3  NodeId id
4  List<NodeId> nList
5  if step = 2k̄ + 1 then
6      sendDataToParent(queue)
7      repeat
8          step ← step - 1
9          getNodes(nList, me, step, 0)
10     until nList.size() > 0 or step = 0
11     doLogging ← false
12 else
13     if step = 0 then
           /* selbst aggregieren                                    */
14         dat.id ← me
15         dat.value ← calcAggregate(me, dataStore)
16         encrypt(predecessor(dat.id, 2k̄ + 1), dat)
17         if me > 0 then queue.add(dat)
18         else reportToBase(dat.value)
19         doLogging ← true
20     if doLogging = false then
           /* Als Zeuge der Knoten in nList: Aggregieren            */
21         forall id in nList do
22             if checkAgg(id) then
23                 dat.id ← id
24                 dat.value ← calcAggregate(id, dataStore)
25                 encrypt(predecessor(dat.id, 2k̄ + 1), dat)
26                 queue.add(dat)
27         sendDataToChildren(queue)
28         doLogging ← true
29     else
           /* Protokollieren, verschlüsseln und weiterleiten        */
30         saveToLog(dat)
31         encrypt(predecessor(dat.id, 2k̄ + 1), queue)
32         sendDataToParent(queue)
33         doLogging ← false
34         if step ≤ 2k̄ + 1-getHeight(me) then step ← 0
35         step ← step - 1
```

Prozedur `EsawnNR.getNodes(`*nList, child, h*`)`

Eingabe : List<NodeId> nList, NodeId child, int h
Ausgabe : List<NodeId> nList

```
   /* Lokale Deklarationen                                          */
 1 List<NodeId> childList
   /* zuerst Knotenliste leeren                                     */
 2 if h = 0 then
 3     nList.clear()
   /* zu step passende Knoten der Liste hinzufügen                  */
 4 if isLeaf(child) = true and step = 2k̄ + 1 then
 5     nList.add(child)
 6 if isLeaf(child) = false and h = step then
 7     nList.add(child)
   /* rekursives Durchlaufen des Aggregationsbaums                  */
 8 if h < step then
 9     childList ← getChildList(child)
10     forall child in childList do
11         getNodes(nList, child, step, h+1)
```

Prozedur `EsawnNR.initAlarm(`*list*`)`

Eingabe : List<DataElement> list

```
   /* Lokale Deklarationen                                          */
 1 DataElement dat
 2 NodeId id
 3 NodeId corruptedId
 4 corruptedId ← dat.id
 5 forall dat in list do
       /* prüfen, ob corruptedId korrumpiert ist                    */
 6     if dat.value ≠ getMajority(list) then
 7         id ← dat.id
           /* Beweis an alle zuständigen Knoten senden              */
 8         while id ≠ me do
 9             sendProof(id, corruptedId, dat.id)
10             sendAlarm(corruptedId)
11             id ← getParent(id)
12     corruptedId ← getParent(corruptedId)
```

Prozedur `EsawnNR.sendAlarm(`*corruptedId*`)`

Eingabe : NodeId corruptedId

```
   /* Lokale Deklarationen                                          */
1  NodeId id
2  int i
3  id ← getParent(me)
4  for i ← 1 to 2k̄ + 1 do
5  |  sendAlarmNoProof(id, corruptedId)
6  |  id ← getParent(id)
7  |  if id = 0 exit for loop
```

Prozedur `EsawnNR.receiveAlarm(`*from, corruptedId*`)`

Eingabe : NodeId from, NodeId corruptedId

```
   /* Lokale Deklarationen                                          */
1  NodeId id
2  int reportCount
3  int i
4  DataElement dat1
5  DataElement dat2
   /* Meldung zu den bisher empfangenen Meldungen hinzufügen        */
6  dat.id ← from
7  dat.value ← corruptedId
8  corruptedNodes.add(dat)
   /* Prüfen, ob mindestens k̄ + 1 von 2k̄ + 1 hintereinanderliegenden
      Knoten den gleichen Betrug gemeldet haben                     */
9  forall dat1 in corruptedNodes do
10 |  reportCount ← 0
11 |  forall dat2 in corruptedNodes do
12 |  |  for i ← 1 to 2k̄ + 1 do
13 |  |  |  if predecessor(dat1.id, i) = dat2.id then
14 |  |  |  |  reportCount ← reportCount +1
15 |  if reportCount ≥ k̄ + 1 then
   |  |  /* Knoten mit ID dat1.value ist korrumpiert                */
16 |  |  sendAlarm(dat1.value)
```

EsawnNR.receiveAlarm()

EsawnNR.receiveAlarm() wird beim Empfang einer Nachricht aufgerufen, die durch *sendAlarmNoProof()* versandt wurde. Da hier keine Überprüfung mittels Schlüsselaufdeckung möglich ist, wird die Zahl der Meldungen über einen korrumpierten Knoten ausgewertet. Dazu werden die Meldungen in einer Liste, *corruptedNodes*, erfasst (Zeile 6-8). Wird die gleiche Meldung von mehr als $\bar{k}$ Knoten, die innerhalb von $2\bar{k}+1$ hintereinanderliegenden Knoten liegen, empfangen (Zeile 10-14), ist von der Legitimität der Meldung auszugehen und die Nachricht entsprechend weiter zu verbreiten (Zeile 16).

4.2 GloMoSim

Wie die meisten realen Netzwerke nutzt GloMoSim ein Schichtenmodell [33] und ermöglicht so eine realitätsnahe Simulation. Die Implementierung erfolgte in der Programmiersprache Parsec [34], einer C-basierten Sprache die besonders für Simulationsanwendungen geeignet und in der das GloMoSim-Framework geschrieben ist.

GloMoSim erwartet Informationen zur Netzwerkstruktur und den Startzuständen der Netzwerkknoten in einer Konfigurationsdatei. Um diese Datei nicht für jedes zu simulierende Netzwerk neu erstellen zu müssen, entstand ein in C geschriebenes Setup-Programm. Dieses erstellt aus den Eingabedaten Netzwerkgröße n, Verzweigungsgrad des Aggregationsbaums δ, Anteil korrumpierter Knoten β sowie den ESAWN-Parametern $k/\bar{k}$ und p ein zufallsgeneriertes Netzwerk. Dieses Netzwerk wird als Eingabe für den Simulationslauf genutzt.

Während eines Simulationslaufes werden eine Reihe von Informationen erfasst und protokolliert. So sind nach Beendigung die Zahl der ausgetauschten Netzwerkpakete, der durchgeführten Aggregationen und Kryptooperationen, eventuelle Alarmmeldungen und erfolgreiche Betrugsversuche sowie der Gesamtenergieverbrauch einsehbar.

Zur Berechnung des Energieverbrauchs wurden auf Daten der MICA2 Sensorknotenplattform [12] und des Sensorknoten-Betriebsystems TinyOS [36] zurückgegriffen. Der Versand eines Netzwerkpakets mit 30 Byte Nutzdaten kostet dort $245\mu As$, Ver- und Entschlüsseln mit RC5 $2,6\mu As$ [35]. Alle wichtigen Simulationsdaten sind in Tabelle 4.1 aufgeführt.

Relativ schnell zeigte sich der enorme Rechenaufwand, der bei großen Netzwerken mit tausenden Sensorknoten auftritt und zu langen Laufzeiten der Simulation führt. Dies führte zu der Idee, die Simulation auf Anwendungs-, Transport- und Netzwerkschicht zu beschränken, da von der Sicherungsschicht kein relevanter Einfluss auf die erfassten Simulationsdaten zu erwarten ist. Protokolliert werden je Knoten die Zahl und Größe der verschickten ESAWN-Datenpakete auf Netzwerkebene und die Zahl der durchgeführten Aggregationen, Crypto- und Vergleichsoperationen. Der Wegfall von CSMA/CA hat keinen Einfluss auf diese Daten.

Zur Umsetzung wurden einige Änderungen am GloMoSim-Framework durchgeführt und die nicht benötigten Schichten deaktiviert. Dazu wurde das Programm so modifiziert, dass die Nachrichten zwischen zwei simulierten Knoten direkt auf der Netzwerkschicht ausgetauscht werden. Die Sicherungsschicht der Simulation wird so umgangen.

Sensorknotenplattform	MICA2
Länge eines Nachrichtenpakets	56 Bytes
Länge der Nutzdaten	30 Bytes
Übertragungsgeschwindigkeit	38,4 kBit/s
Energieverbrauch je Nachrichtenpaket	$245\mu As$
Energieverbrauch je Cryptooperation	$1{,}3\mu As$
Crytoalgorithmus (Blocklänge)	RC5 (8 Byte)

Tabelle 4.1: Wichtige Simulationsparameter, Quellen: [4; 3; 36; 35]

Diese Änderungen sorgten für einen gut zehnfach schnelleren Simulationsablauf ohne die Ergebnisse zu beeinflussen. Die zur Probe durchgeführten Simulationen mit CSMA/CA belegen dies und sind im Anhang in Abschnitt A.2 aufgeführt.

4.3 SecuSim

SecuSim war anfangs nur dafür gedacht, die „Wahrscheinlichkeit einer korrekten Aggregation“ für bestimmte Netzwerkkonfigurationen durch Simulation zu bestimmen. Diese Aufgabe führte auch zu der Namensgebung „SecuSim“. Im Laufe der Zeit wurde das Programm dann um eine Vielzahl von Ausgabemöglichkeiten erweitert und gibt mittlerweile eine Vielzahl statistischer Daten aus, die über den Grad der erreichten Authentizität hinausgehen.

SecuSim abstrahiert von der detailreichen Simulation bei GloMoSim und erlaubt es so in kurzer Zeit diese statistische Daten zu erhalten. Da SecuSim komplett auf die Simulation des Protokollablaufs und die zugehörigen Netzwerkpakete verzichtet und lediglich gemäß den Eingabeparametern Aggregationsbäume generiert und statistische Daten hierzu erstellt, benötigt ein Durchlauf mit tausenden Sensorknoten nur wenige Sekunden.

Im Einzelnen sammelt SecuSim Informationen zu Zahl der ausgetauschten Netzwerkpakete, der Zahl der durchgeführten Aggregationen und Kryptooperationen, dem Gesamtenergieverbrauch und der Wahrscheinlichkeit einer korrekten authentischen Aggregation.

4.3.1 Energieverbrauch

Um den Energieverbrauch zu bestimmen, werden die Zahl der versandten Pakete sowie die Zahl der notwendigen Kryptooperationen erfasst. Zusammen mit den Verbrauchsdaten der MICA2-Sensorplattform, die auch in der GloMoSim-Implementierung verwendet wurden, ergibt sich der Gesamtenergieverbrauch.

4.3.2 Speicherverbrauch

SecuSim analysiert den Speicherverbrauch nach verschiedenen Kriterien. Zum einen wird der mittlere Speicherverbrauch pro Knoten berechnet. Da im Aggregationsbaum jedoch ein großer Teil Blattknoten mit einem Speicherverbrauch von 1 sind, ist dieser Wert nur beschränkt aussagekräftig. Besser ist der mittlere Speicherverbrauch pro Aggregationsknoten. Um einen Eindruck über mögliche besonders hohe

Gesamtpaket			
TinyOS-Header	Datum 1	Datum 2	Datum 3
...	Byte 1-10	Byte 11-20	Byte 21-30

Datum									
ID		Wert				Sequenznummer			
01	02	03	04	05	06	07	08	09	10

Tabelle 4.2: ESAWN-Paketformat in TinyOS

Speicheranforderungen zu erhalten, wird zusätzlich der maximale Speicherverbrauch der auf einem Knoten auftritt, ausgegeben.

All dies erfordert jedoch zuerst die Berechnung des konkreten Speicherverbrauchs für jeden einzelnen Knoten. Da die Bäume durch die zufällige Generierung in aller Regel keine Gleichmäßigkeit aufweisen, kann auch nicht auf konkrete Formeln zurückgegriffen werden.

SecuSim betrachtet daher für jeden Aggregationsknoten den jeweils relevanten Teilbaum um den Speicherverbrauch zu erhalten. Der relevante Teilbaum umfasst alle Kindknoten des Aggregationsknotens A bis zu einer Tiefe von $k + 2$ bzw. $2\bar{k} + 2$. Er enthält damit sämtliche Knoten von denen A Werte erhält. Die Nachrichten, die A von diesen Knoten erhalten würde, werden nun erfasst und daraus der maximale Speicherbedarf an Messwerten während eines Messvorgangs bestimmt.

Aus den Daten für jeden Knoten können dann die statisischen Daten wie Minimum, Maximum und mittlerer Speicherverbrauch pro Knoten bestimmt werden.

4.4 TinyOS / MICAz

Abschließend erfolgte die Implementierung auf einer realen Sensorknotenplattform unter TinyOS, welches die Programmiersprache nesC, eine Erweiterung von C, nutzt [36]. Als Hardware kamen MICAz-Sensorknoten zum Einsatz. Die Daten dieses Sensorknotentyps wurden auch als Basis für die Simulationsergebnisse von Energieverbrauch u.ä. verwendet.

Die drei Protokolle, ESAWN, ESAWN-2 und ESAWN-NR, wurden in nesC implementiert und eine entsprechendes Paketformat definiert. Es ermöglicht bei einer Nutzdatengröße von 30 Byte den Transport von bis zu drei ID-Wert-Paaren.

Die Implementierung wurde auf sechs MICAz-Sensorknoten gespielt, denen die TinyOS-IDs 0 bis 5 zugewiesen wurden. Um das Netz mit verschiedenen Aggregationsbäumen zu testen, entstand eine in Java geschriebene Steuerungsapplikation die über die serielle Schnittstelle mit der Senke des Netzwerks (ID = 0) kommuniziert. In der Anwendung können die Zahl der Knoten, die Parameter k beziehungsweise $\bar{k}$ und p und der Aufbau des Aggregationsbaum festgelegt werden. Zusätzlich werden automatisch die benötigten Kommunikationsschlüssel erstellt. Alle Daten werden mit einem Befehl an die Senke übertragen. Diese verteilt die Informationen im Netz, so dass alle Knoten die benötigten Informationen über den Baumaufbau, Protokollparameter und ihre zugewiesenen Schlüssel erhalten.

Nach dieser Initialisierung des Netzwerks kann mit einem weiteren Befehl ein Messvorgang gestartet werden.

Zu Demonstrationszwecken wurde der Ablauf des Protokolls auf den Sensorknoten künstlich gebremst und der Fortschritt des Protokolls mit Hilfe der drei farbigen LEDs des Sensorknotens visualisiert. Eine grüne LED zeigt dabei einen initialisierten und im Ruhezustand befindlichen Knoten an. Eine gelbe LED zeigt Protokollaktivität an. Durch Blinken der roten LED werden verschiedene Fehler angezeigt. Stellt ein Knoten bei Durchführung des Mehrheitsentscheides eine Abweichung fest, wird dies durch dauerhaftes Leuchten der roten LED signalisiert.

Im rechten Teil des Anwendungsfensters der Steuerapplikation kann außerdem der Protokollablauf verfolgt werden. Hier werden alle versandten Pakete und die durchgeführten Aggregationen und Mehrheitsentscheide protokolliert.

5. Evaluierung

Ziel des Kapitels: In diesem Kapitel werden verschiedene Simulationsergebnisse vorgestellt, die die Effizienz und Leistungsfähigkeit der ESAWN-Protokolle darstellen und veranschaulichen sollen. Während sich der erste Teil mit dem Energieverbrauch und Speicherverbrauch beschäftigt, steht im zweiten Teil der erzielbare Grad an Sicherheit im Vordergrund. Dabei wird aufgezeigt, unter welchen Bedingungen der Einsatz von ESAWN-2 und ESAWN-NR sinnvoll ist.

Aufbau des Kapitels: Zuerst werden in Abschnitt 5.1 die Randbedingungen, unter denen simuliert wurde, beschrieben. Es folgt mit Abschnitt 5.2 eine Betrachtung des Energieverbrauchs der ESAWN-Protokolle. Analog dazu geht Abschnitt 5.3 auf den Speicherverbrauch der Protokolle ein. Betrachtungen zur „Wahrscheinlichkeit korrekter Aggregation“ finden sich in Abschnitt 5.4. Abgeschlossen wir mit einer Zusammenfassung in Abschnitt 5.5.

5.1 Simulationsdaten

Die Daten stammen aus der GloMoSim- und der SecuSim-Implementierung. Soweit es die Laufzeit ermöglichte, wurden direkt Daten der GloMoSim-Simulation verwendet. In anderen Fällen wurde auf die schneller verfügbaren Ergebnisse von SecuSim zurückgegriffen, die durch einzelne Tests mit der GloMoSim-Implementierung gegengeprüft wurden. Dies ermöglichte es, in vertretbarer Zeit, eine Vielzahl möglicher Netzwerkkonfigurationen zu untersuchen und zuverlässige Werte zu erhalten. Details dazu finden sich im Anhang, Abschnitt A.3.

Generell wurden Netzwerke mit $1000 \leq n \leq 10000$ Knoten betrachtet in denen ein Aggregationsbaum mit Verzweigungsgrad $2 \leq \delta \leq 4$ aufgebaut wurde. Dabei wurden unterschiedliche Anteile korrumpierter Knoten $\beta \in \{0; 0{,}01; 0{,}05; 0{,}1; 0{,}2\}$ angenommen.

Zur Kurzbezeichnung werden wieder die in Tabelle B.1 eingeführten Variablennamen verwendet.

5.2 Energieverbrauch

Geringer Energieverbrauch ist eines der Hauptkriterien zur Beurteilung eines Protokolls in drahtlosen Sensornetzen. Daher liegt die naheliegendste Möglichkeit die ESAWN-Protokolle zu bewerten, in der Auswertung des benötigten Energieverbrauchs.

Obwohl in diesem Kapitel von „Energieverbrauch“ die Rede ist, sind alle folgenden Angaben – auch wenn von Energieverbrauch die Rede ist – Ladungsmengen in As. Der Energieverbrauch E verhält sich proportional zur Ladungsmenge Q. Es gilt $E = Q \cdot U$. Die Spannung U ist bei allen Vergleichen gleich, da immer die Daten eines MICA2-Sensorknotens die Grundlage darstellen. Damit ist ein Vergleich der Ladungsmengen anstelle des Energieverbrauchs legitim.

Die in den Abschnitten 5.2.1 und 5.2.2 vorgestellten Ergebnisse berücksichtigen dabei jedoch nicht den Energieverbrauch, der für die Aggregation der Messdaten anfällt. Die ESAWN-Protokolle sind, was die Aggregationsfunktion betrifft, flexibel. Das bedeutet, sie funktionieren generell mit allen denkbaren Aggregationsfunktionen. Im Anhang in Abschnitt A.1 wird daher gezeigt, dass die Energiekosten der Aggregationen im Vergleich zum Gesamtenergieverbrauch des Protokolls vernachlässigbar gering sind, selbst für komplexe Aggregationsfunktionen.

5.2.1 Absolute Kosten

Die Grafiken in Abbildung 5.1 visualisieren den absoluten Energieverbrauch, den ein vollständiger Messvorgang mit den Protokollen ESAWN, ESAWN-2 und ESAWN-NR pro Sensorknoten im Durchschnitt benötigt.

Unter einem vollständigen Messvorgang wird hier und im Folgenden ein kompletter Aggregationsvorgang von den Blattknoten des Aggregationsbaums bis hin zur Senke bezeichnet.

Diese Auswertung beschränkt sich, der Übersichtlichkeit halber, auf einen Verzweigungsgrad von $\delta = 2$, ausführlichere Analysen, auch mit anderen δ, befinden sich im Abschnitt 5.2.2.

Betrachtet man zuerst jede der drei Grafiken für sich, lässt sich gut der Mehrbedarf an Energie erkennen, der mit größerem k bei ESAWN bzw. größerem $\bar{k}$ bei ESAWN-2 und ESAWN-NR entsteht. Dies ist zum einen darauf zurückzuführen, dass mehr Daten ver- und entschlüsselt werden müssen. Zum anderen, und das ist der Großteil des Mehrbedarfs, an den größeren Nachrichten und damit der größeren Paketzahl, die zwischen den Knoten ausgetauscht werden müssen.

Betrachtet man nun alle drei Grafiken im Vergleich, so sieht man die konstante Steigerung des Energiebedarfs von ESAWN auf ESAWN-2 und von ESAWN-2 auf ESAWN-NR bei ansonsten gleicher Konfiguration. Bei allen drei Protokollen liegt der Energieverbrauch in $O(1)$, er ist also unabhängig von der Netzwerkgröße.

Eine Ausnahme gibt es bei großen $\bar{k}$ und gleichzeigt kleinen Netzgrößen n. Hier liegt der Energieverbrauch bei niedriger, als zu erwarten wäre. Dies liegt daran, dass bei kleinen Netzwerken die Aggregationspfade zu kurz sind, um die benötigte Anzahl von $2\bar{k}+1$ Zeugen einzusetzen. Auf diese Problematik wird in Abschnitt 5.2.2 ausführlich eingegangen.

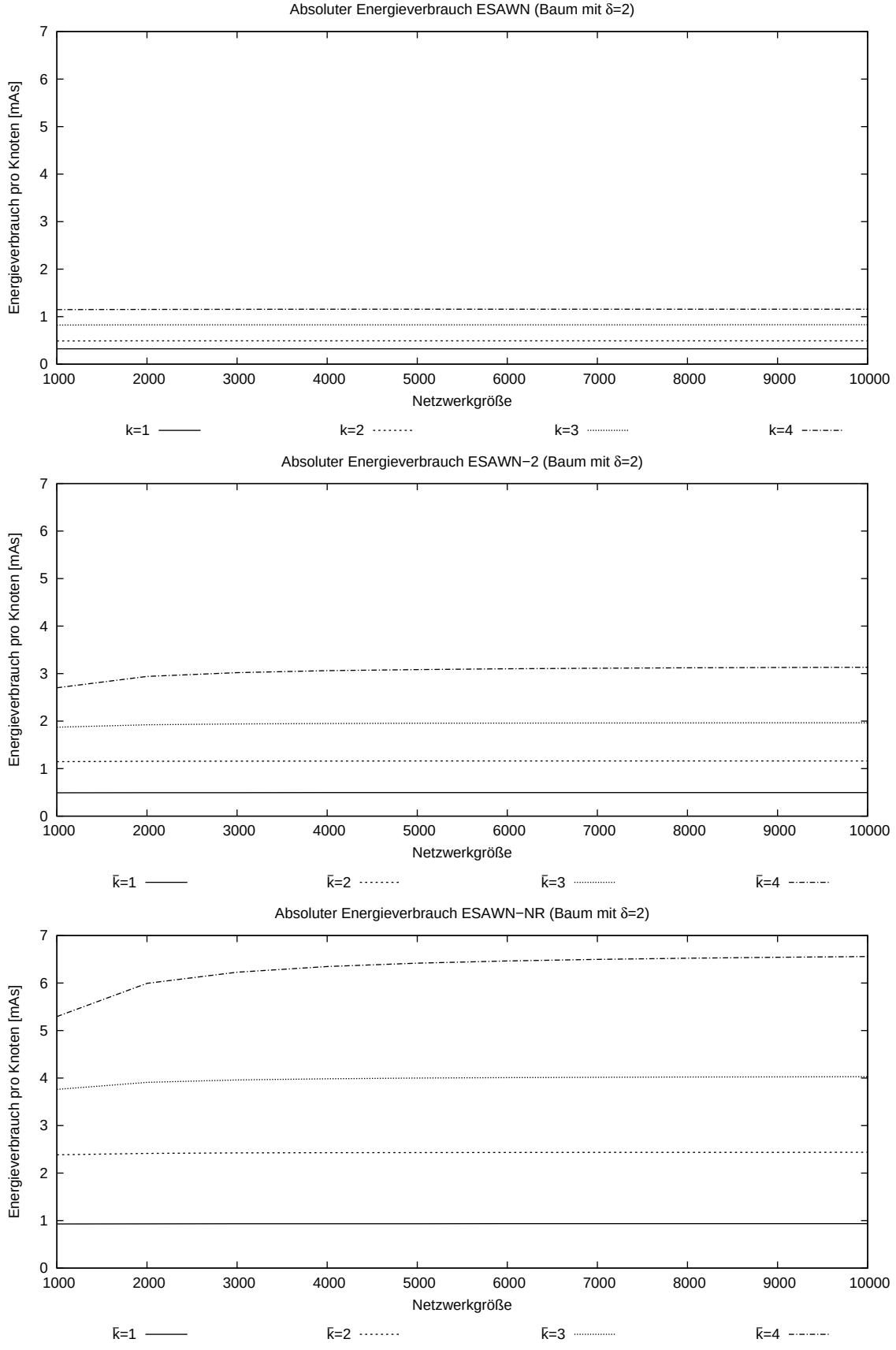


Abbildung 5.1: Absolute Energiekosten pro Knoten bei $\delta = 2$ und $k = \bar{k} = 1..4$

5.2.2 Einsparpotential der sicheren Aggregation

Ein wesentlicher Vorteil der ESAWN-Protokolle ist die erzielte Energieeinsparung, ohne völlig auf Authentizität zu verzichten. Die folgenden Simulationsergebnisse belegen, wieviel Energie sich mit ESAWN, ESAWN-2 und ESAWN-NR tatsächlich einsparen lässt.

Die folgenden Diagramme stellen den Energieverbrauch in Relation zu einem fiktiven Protokoll NOAG dar und enthalten außerdem den Verbrauch eines zweiten fiktiven Protokolls AGGN. Diese beiden Protokolle unterstützen die folgende Evaluierung.

NOAG

NOAG steht dabei für ein simples aber sicheres Transportprotokoll, bei dem jeder Blattknoten seinen Wert für die Senke verschlüsselt und direkt an sie versendet. Damit erzielt NOAG Authentizität, nutzt jedoch keine Aggregation. Ein Protokoll zur sicheren Aggregation sollte weniger Energie verbrauchen als NOAG, da es sonst keine Ersparnis zu einer nicht-aggregierenden Kommunikation bringt. Dies ist auch der Grund, weshalb die folgenden Ergebnisse in Relation zu NOAG dargestellt werden. So lässt sich sofort die Einsparung ablesen, die das jeweilige Protokoll im Vergleich zu einer sicheren, nicht-aggregierenden Kommunikation erzielt.

AGGN

AGGN steht für das zweite Protokoll, welches eine simple Aggregation an jedem Aggregationsknoten im Netzwerk durchführt, allerdings ohne die Daten dabei zu verschlüsseln. AGGN stellt damit eine untere Schranke dar, da AGGN den Mindestaufwand eines aggregierenden Kommunikation darstellt. Ein Protokoll zur sicheren Aggregation kann diesen nicht unterschreiten, da es mehr leisten muss, als die reine Aggregation.

Übersicht

Die nächsten Diagramme zeigen das Einsparpotential für Sensornetze mit $1000 \leq n \leq 10000$ Knoten und $\delta \in \{2, 3, 4\}$ sowie k, $\bar{k} \in \{1, 2, 3, 4\}$. Sie gliedern sich wie folgt in die 6 nächsten Abbildungen:

	$k/\bar{k} = 1$	$k/\bar{k} = 2$	$k/\bar{k} = 3$	$k/\bar{k} = 4$
$\delta = 2$	Abb. 5.2		Abb. 5.3	
$\delta = 3$	Abb. 5.4		Abb. 5.5	
$\delta = 4$	Abb. 5.6		Abb. 5.7	

Simulation mit $\delta = 2$ und $k = \bar{k} = 1$

Betrachtet man zunächst das obere Diagramm in Abbildung 5.2, lässt sich sehr deutlich die Ersparnis erkennen, welche die ESAWN-Protokolle im Vergleich zu NOAG bei $\delta = 2$ und $k = \bar{k} = 1$ bieten. ESAWN benötigt, abhängig von der Netzwerkgröße, nur etwa 10% der Energie von NOAG. Auch ESAWN-2 und ESAWN-NR bieten mit ca. 15% und 30% deutliche Einsparungen. Erwartungsgemäß fällt die Ersparnis bei

ESAWN-2 geringer als bei ESAWN aus. Im Vergleich zu ESAWN, bei dem nur ein Zeuge benötigt wird, ist, bedingt durch die Mehrheitsentscheide bei denen $2\bar{k}+1 = 3$ Knoten beteiligt sind, hier deutlich mehr Kommunikation notwendig.

Die Kurve, die ESAWN-NR beschreibt, liegt wiederum nochmals signifikant über der Kurve von ESAWN-2. Hier wird zwar die gleiche Menge an Daten übertragen, dies geschieht jedoch über Umwege, die zudem auch die Möglichkeiten der Paketaggregation einschränken. Bei ESAWN und ESAWN-2 kann ein Knoten stets alle Daten sammeln um sie dann gebündelt an den Vaterknoten zu senden. Er führt damit seine Aufgabe als Aggregationsknoten und die als Zeugen parallel aus. Bei ESAWN-NR ist dies nicht möglich. Ein Knoten muss dort seine verschiedenen Aufgaben als Teilnehmer von $2\bar{k}$ Mehrheitsentscheiden und einer eigenen Aggregation streng sequentiell ausführen, da die verschiedenen Schritte aufeinander aufbauen. Jeder der beteiligten Knoten braucht alle Daten eines Schrittes, um diesen abzuschließen. Er kann erst dann mit dem nächsten Schritt fortfahren. Bei gleicher zu übertragender Datenmenge, führt dies zu einem deutlichen Mehrverbrauch an Energie, bedingt durch die höhere Zahl an zu versendenden Datenpaketen.

Bleibt noch der Grund für die insgesamt leicht abfallenden Kurven zu klären. Die Energieersparnis ist in großen Netzwerken und gleichem δ offensichtlich größer als bei kleinen Netzwerken. Dies liegt daran, dass hier die Aggregationspfade vom Blattknoten zur Senke länger sind, als in kleinen Netzwerken und somit die Vorteile der Datenaggregation deutlicher zum Tragen kommen, als dies auf kurzen Wegen der Fall wäre.

Simulation mit $\delta = 2$ und $k = \bar{k} = 2$

Im unteren Diagramm der Abbildung ist die gleiche Situation jedoch mit $k = \bar{k} = 2$ dargestellt. Hier sind die monoton fallenden Kurven noch besser erkennbar. Genauso wie der Mehraufwand der ESAWN-2 und ESAWN-NR Protokolle im Vergleich zu ESAWN. Bemerkenswert ist, dass bereits mit $\bar{k} = 2$ die Ersparnis bei ESAWN-NR im Vergleich zu NOAG auf grade einmal 20%-40% zurückgegangen ist. Damit nähert sich ESAWN-NR in dieser Konfiguration bereits deutlich an den Energieaufwand von NOAG an.

Simulation mit $\delta = 2$ und $k = \bar{k} = 3$ oder $k = \bar{k} = 4$

In der nächsten Abbildung 5.3 wurden noch höhere Werte für k und $\bar{k}$ gewählt. Im oberen Diagramm gilt $k = \bar{k} = 3$, im unteren $k = \bar{k} = 4$. Schon im ersten Fall ist klar zu erkennen, dass die Ausführung von ESAWN-NR jetzt mehr Energie kosten würde, als das Vergleichsprotokoll NOAG. Im unteren Diagramm hat die Kurve sogar den Skalenbereich verlassen und liegt bei etwa 160% bis 180%. Dies bedeutet, dass ein nicht-aggregierender Datentransport hier günstiger ist und zudem den gleichen Sicherheitsansprüchen genügt. Ein Einsatz von ESAWN-NR in Netzwerken mit Aggregationsbäumen dieser Struktur kommt daher nur für $\bar{k} \leq 2$ in Frage.

Simulation mit $\delta > 2$

Die bisherigen Betrachtungen gelten für $\delta = 2$. Was bei einem höheren Verzweigungsgrad passiert, veranschaulichen die nächsten Diagramme.

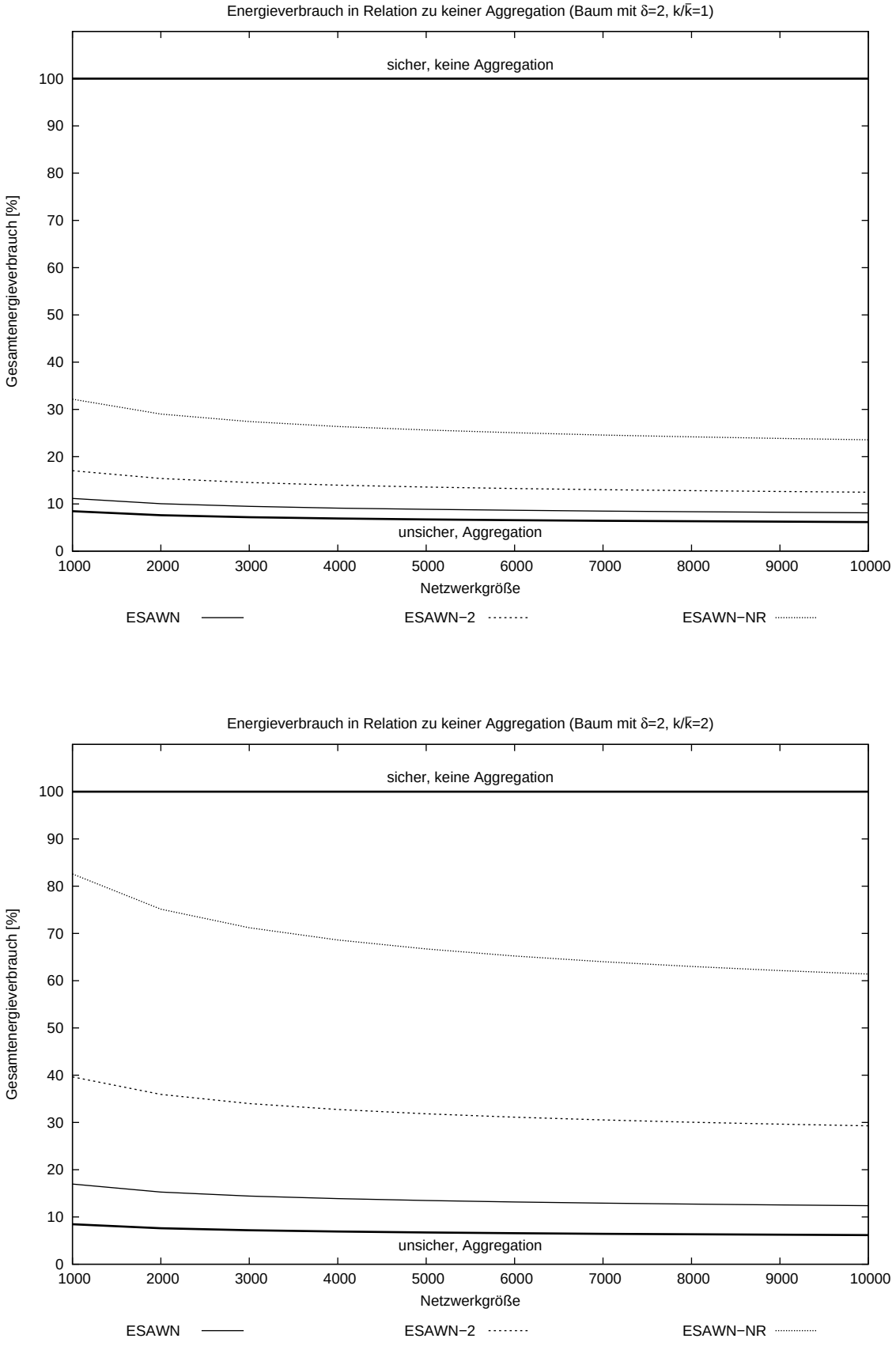


Abbildung 5.2: Relative Energiekosten in einem Aggregationsbaum mit $\delta = 2$ und $\bar{k} = 1..2$

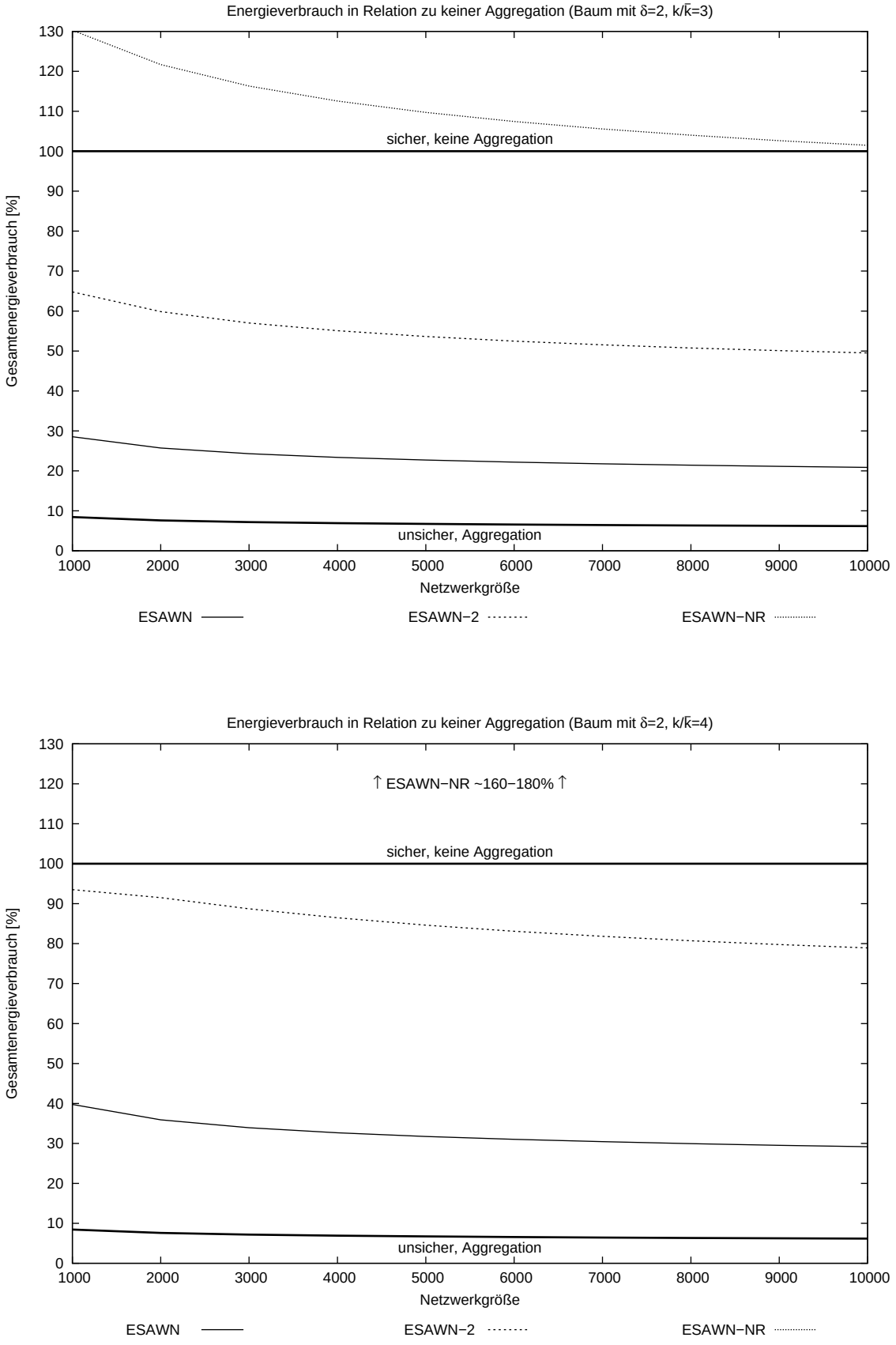


Abbildung 5.3: Relative Energiekosten in einem Aggregationsbaum mit $\delta = 2$ und $\bar{k} = 3..4$

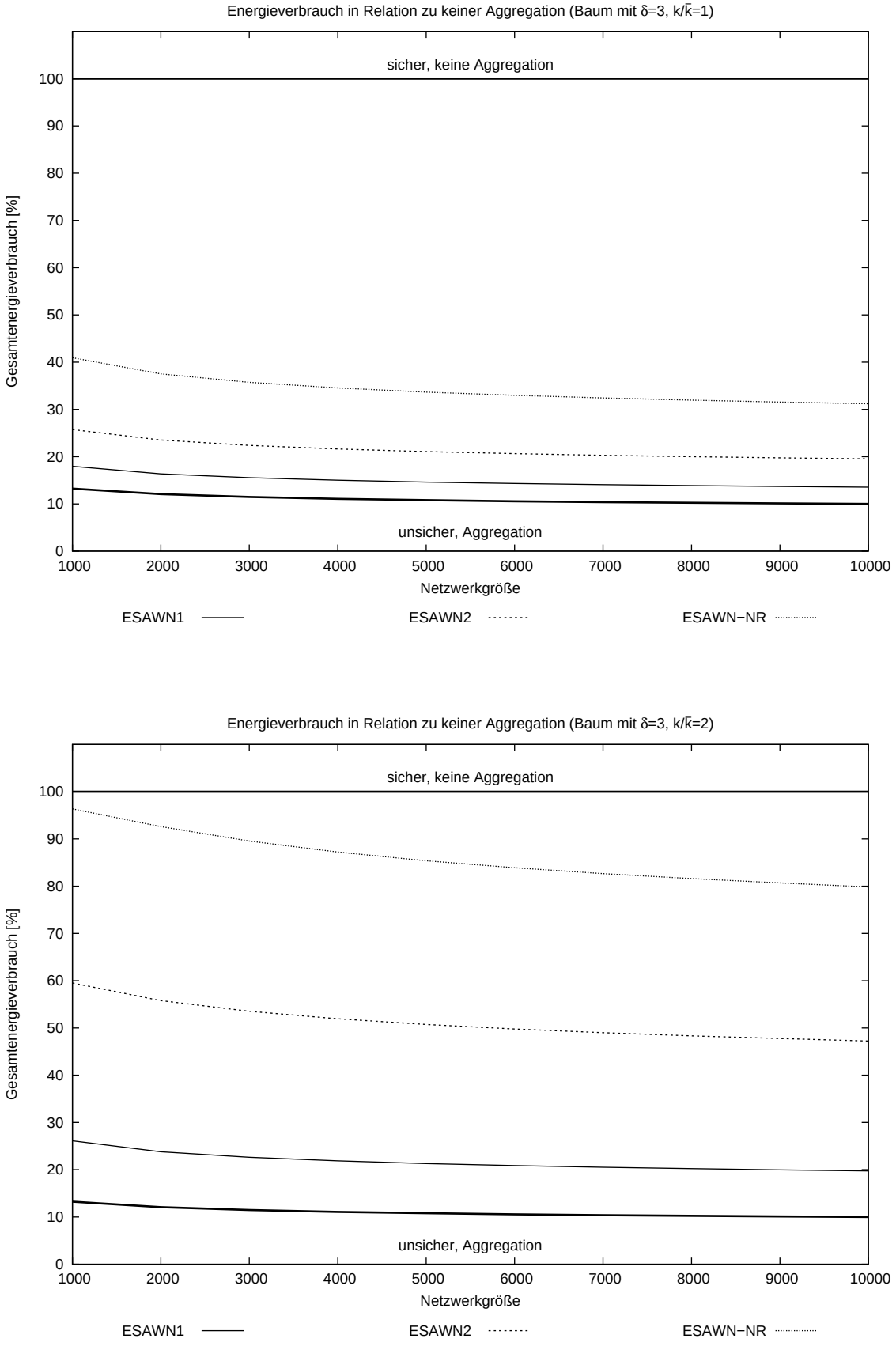
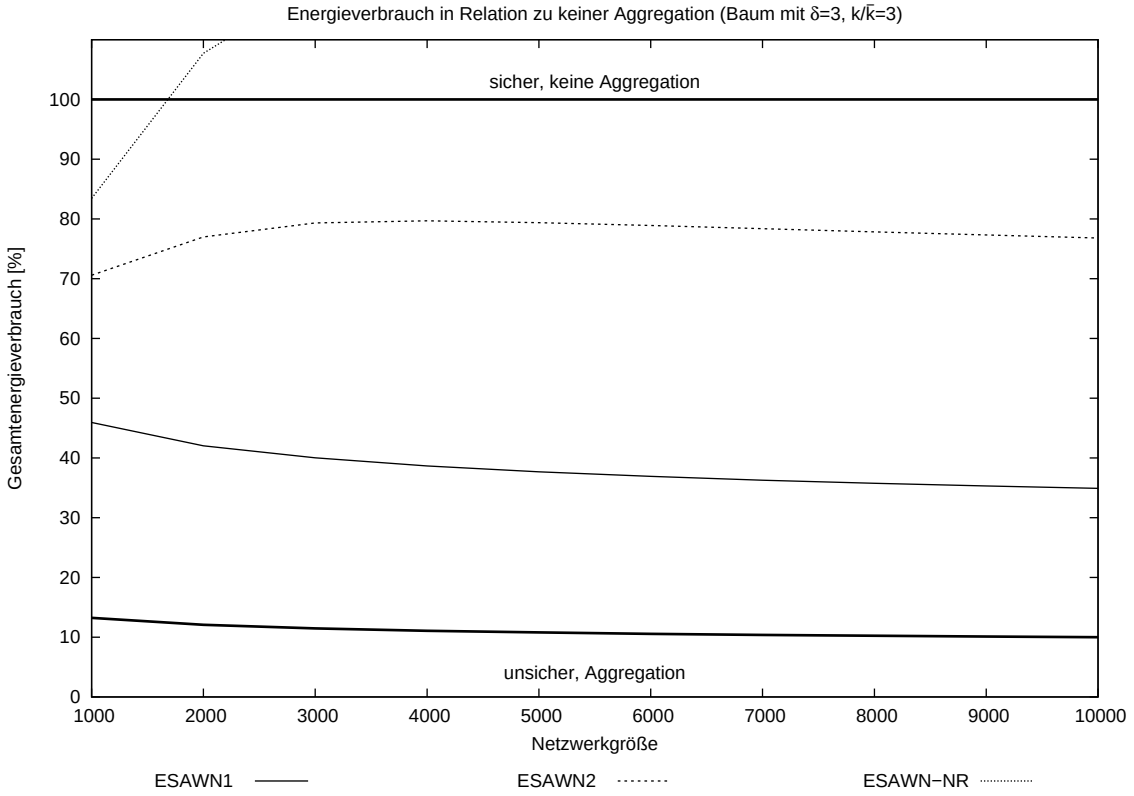


Abbildung 5.4: Relative Energiekosten in einem Aggregationsbaum mit $\delta = 3$ und $\bar{k} = 1..2$



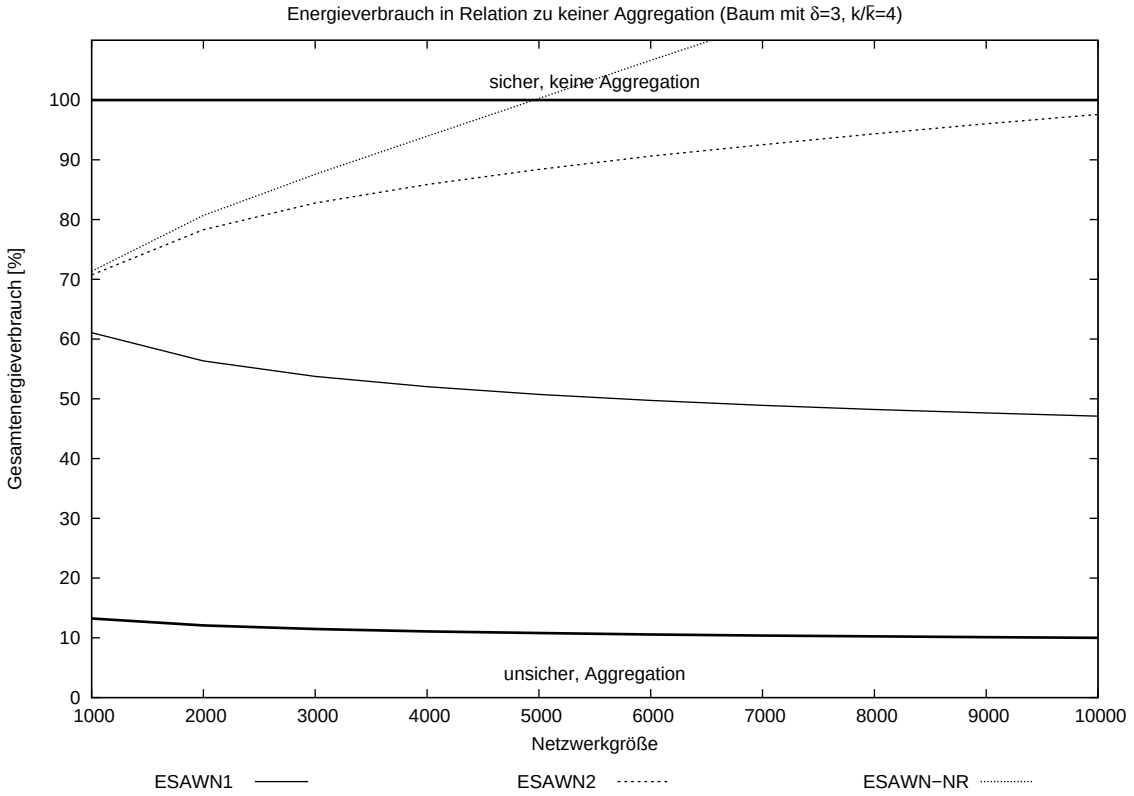


Abbildung 5.5: Relative Energiekosten in einem Aggregationsbaum mit $\delta = 3$ und $\bar{k} = 3..4$

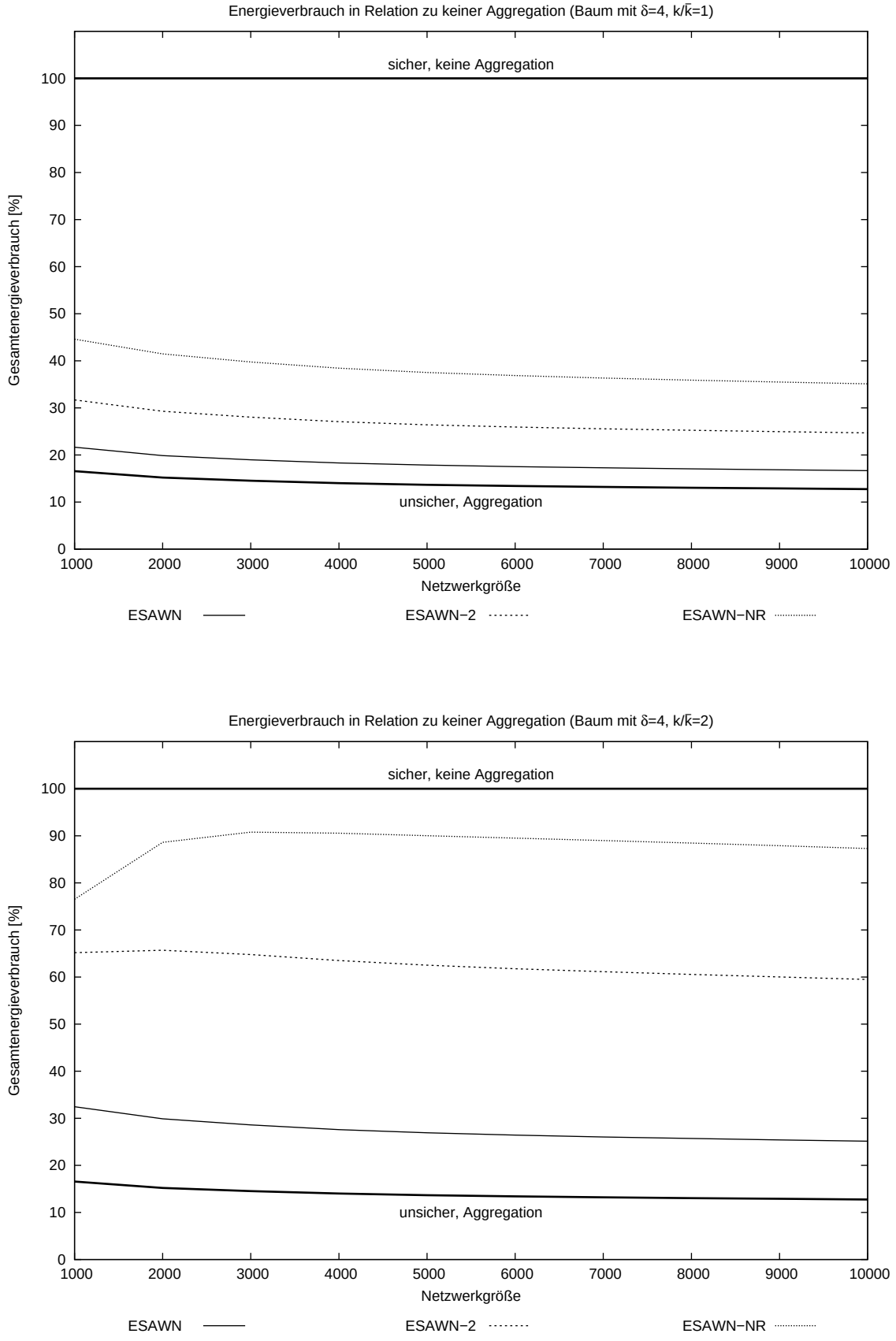


Abbildung 5.6: Relative Energiekosten in einem Aggregationsbaum mit $\delta = 4$ und $\bar{k} = 1..2$

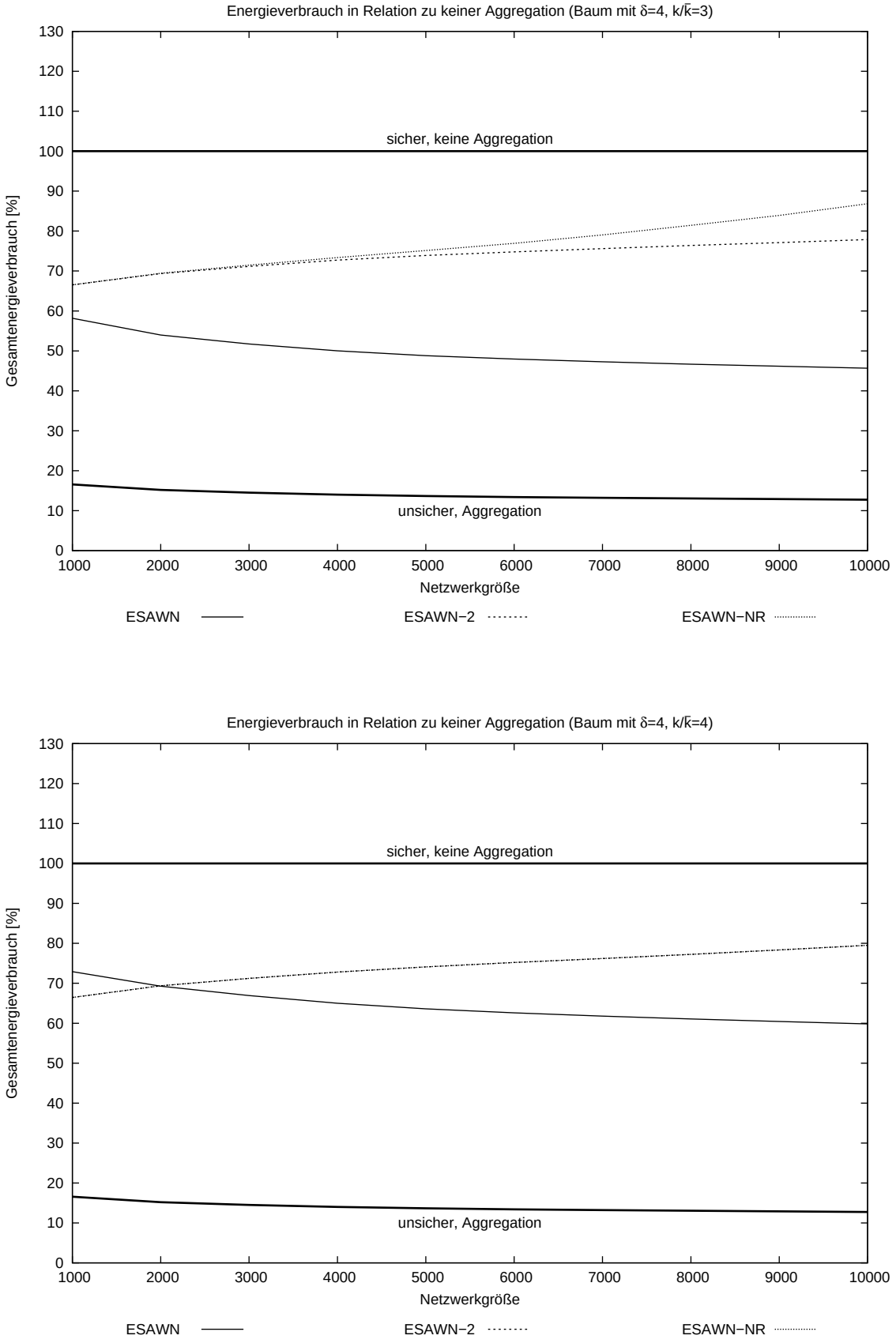


Abbildung 5.7: Relative Energiekosten in einem Aggregationsbaum mit $\delta = 4$ und $\bar{k} = 3..4$

Als erstes werden die Diagramme verglichen, die sich jeweils oben in den Abbildungen 5.2, 5.4 und 5.6 befinden. Hier gilt bei gleichem k und $\bar{k}$ ein Verzweigungsgrad von $\delta = 2$, $\delta = 3$ und $\delta = 4$. Mit steigendem δ wird die Einsparung im Vergleich zu NOAG geringer. Die Ursache ist dieselbe wie die der abfallenden Kurven. Durch ein größeres δ verkürzen sich bei gleicher Knotenzahl die Aggregationspfade. Die Vorteile der Datenaggregation reduzieren sich dadurch. Deshalb sind davon auch nicht nur die ESAWN-Protokolle betroffen, sondern auch das Vergleichsprotokoll AGGN, dessen Kurve auch höher liegt.

An dieser Stelle ist es wichtig, darauf hinzuweisen, dass die höheren Kurven keinen höheren Energieverbrauch bedeuten. Im Gegenteil, bei gleicher Knotenzahl und größerem δ fallen die Kosten der ESAWN-Protokolle geringer aus. Durch ein größeres δ werden bei einer Aggregation mehr Daten zusammengefasst und es müssen weniger Informationen übertragen werden. Gleichzeitig verkürzen sich die Kommunikationswege, da die Höhe des Aggregationsbaums bei größerem δ und gleicher Knotenzahl n kleiner wird. Lediglich die Einsparung gegenüber NOAG reduziert sich und dies führt zu höheren Kurven in diesen Diagrammen.

Sonderfall bei zu großem k / $\bar{k}$

Abschließend verbleiben noch einige Diagramme, die aus der Reihe zu fallen scheinen. Die beiden ersten dieser Art sind die Diagramme in Abbildung 5.5. Hier gilt $\delta = 3$ und $k = \bar{k} = 3$ bzw. $k = \bar{k} = 4$. Hier verlaufen die Kurven teilweise ansteigend und auch der gleichmäßige Abstand zwischen den Kurven von ESAWN, ESAWN-2 und ESAWN-NR ist hier nicht gegeben.

Der Grund hierfür liegt in der Implementierung der ESAWN-Protokolle und lässt sich am Beispiel des Kurvenverlaufs von ESAWN-2 oben in Abbildung 5.5 erklären. Die Kurve steigt erst leicht an, um dann bei einer Netzwerkgröße von etwa 3000-4000 Knoten wieder in gewohnter Weise abzufallen.

Offenbar kommt es hier bei kleinen Netzwerken $n \leq 3000$ Knoten zu geringeren Energiekosten, als dies aufgrund der bisherigen Ergebnisse anzunehmen wäre. Das liegt an den kurzen Aggregationspfaden in solchen Netzwerken. Bei $n = 1000$ und $\delta = 3$ beträgt die mittlere Höhe des Aggregationsbaums $h = 5{,}92 \approx 6$ (zur Berechnung siehe Abschnitt 2.3.1). Bei $n = 3000$ und $\delta = 3$ gilt $h = 6{,}92 \approx 7$. Abzüglich Blatt- und Wurzelknoten stehen somit 4 bzw. 5 innere Knoten zur Verfügung.

Ein ESAWN-2 Protokollablauf benötigt bei $\bar{k} = 3$ jedoch $2\bar{k} + 1 = 7$ Knoten für einen Mehrheitsentscheid. Da dies hier nicht möglich ist, sendet der Blattknoten seinen Wert direkt an die Senke. Da das direkte Versenden in diesem Fall günstiger ist, sinken die Gesamtkosten. Sobald ausreichend innere Knoten vorhanden sind, wird ESAWN-2 wie gewohnt mit Mehrheitsentscheiden durchgeführt. Dass dies nicht immer sinnvoll ist, zeigen die Kurven die über dem NOAG-Protokoll liegen. Hier wäre das direkte Senden des Messwertes an die Senke sinnvoller, weil kostengünstiger, gewesen.

Im oberen Diagramm der Abbildung 5.7 geht dies soweit, dass bei kleinen Netzwerken mit $n \leq 3000$ Knoten die Kurven von ESAWN-2 und ESAWN-NR sogar aufeinander liegen. Hier gibt es keine Pfade ausreichender Länge mehr und ESAWN-2 und ESAWN-NR verhalten sich wie NOAG, das heißt die Blattknoten senden ihren Messwert direkt an die Senke. Trotzdem wird hier eine Einsparung erzielt, die auf die

Paketaggregation zurückzuführen ist, die ESAWN-2 und ESAWN-NR im Gegensatz zu NOAG durchführen. Die inneren Knoten können bis zu drei Messwerte in einer Nachricht weiterleiten und so Energie sparen.

Im unteren Diagramm der selben Abbildung liegen die Kurven für ESAWN-2 und ESAWN-NR im gesamten Darstellungsbereich aufeinander. Hier ist durchgehend kein vorgesehener Protokollablauf möglich. Die reine Paketaggregation von ESAWN-2 ist hier aus energetischer Sicht sogar kurzzeitig ESAWN überlegen.

Fazit

Bei kleinem $\bar{k}$ ist eine deutlich Energieeinsparung von ESAWN-2 und ESAWN-NR, im Vergleich zu NOAG, erkennbar. Die Einsparung nimmt mit größerem $\bar{k}$ jedoch schnell ab, übersteigt die Kosten von NOAG und wird somit uninteressant für den praktischen Einsatz. Bei noch größerem $\bar{k}$, sinkt der Energiebedarf jedoch wieder, aufgrund zu kurzer Pfadlängen für die benötigte Anzahl Zeugen.

5.3 Speicherverbrauch

Neben dem Energieverbrauch ist der Speicherverbrauch eines Protokolls das zweite wichtige Kriterium für den Einsatz in einem Sensornetz.

Der Speicherverbrauch lässt sich dabei in verschiedene Bereiche aufteilen. Zum einen die Zahl der Messwerte und Aggregate, die ein Sensorknoten während eines Protokolldurchlaufs zwischenspeichern muss. Zum anderen der Speicherverbrauch durch statische Daten, wie Netzwerkstruktur und Kommunikationsschlüssel, die dem Knoten bereits vor Protokollablauf vorliegen. Einen dritten Teil beansprucht der Anwendungscode selbst.

Im Folgenden sollen die beiden ersten Teile genauer betrachtet werden. Der genaue Speicherbedarf hängt auch von der konkreten Implementierung des Protokolls ab. Daher werden hier insbesondere die Aspekte betrachtet, die abhängig von bestimmten Netzwerkparametern skalieren und so zu potentiellen Engpässen bei großen Netzwerken führen können.

Generell wird der maximale Speicherbedarf pro Knoten betrachtet. Aufgrund der Lage eines Knotens (beispielsweise nahe der Senke oder den Blattknoten) ist durchaus auch ein niedrigerer Speicherbedarf denkbar.

5.3.1 NOAG

Beim NOAG Protokoll müssen die Aggregationsknoten die eingehenden verschlüsselten Werte unverändert weiterleiten. Es muss zu jedem Zeitpunkt also höchsten 1 Wert gespeichert werden.

Ein Wissen über die Netzwerkstruktur ist nicht notwendig, lediglich die Blattknoten müssen einen kryptographischen Schlüssel speichern. Maximal somit 1 Schlüssel.

Der Speicherverbrauch liegt somit in $O(1)$ und ist konstant und unabhängig von sämtlichen Netzwerkparametern.

5.3.2 AGGN

Im AGGN Protokoll muss jeder Aggregationsknoten nur die direkt eingehenden Werte (δ Stück) und das daraus berechnete Aggregat speichern. Es ergibt sich daher $(\delta + 1)$ Werte.

Zum korrekten Ablauf muss ein Knoten lediglich seinen Elternknoten und die direkten Kindknoten kennen. Es ergibt sich ein statischer Speicherbedarf von $(\delta + 1)$ Knotenadressen.

Der Speicherverbrauch liegt somit in $O(1)$ bezogen auf die Netzwerkgröße. Er hängt einzig und allein vom Verzweigungsgrad des Aggregationsbaums ab, also $O(\delta)$.

5.3.3 ESAWN

Ein ESAWN-Knoten muss zum einen die normale Aggregation wie beim AGGN-Protokoll durchführen und zum anderen für alle bis zu k Schritte im Aggregationsbaum entfernte Kindknoten Zeuge sein, also deren Aggregation zur Kontrolle selbst durchführen. Dies bedeutet, dass er Werte von seinen direkten Kindern, aber auch von alle anderen bis zu $k + 1$ Schritte entfernten Kindknoten empfangen und speichern muss. Insgesamt also $(\sum_{h=0}^{k+1} \delta^h)$ Werte.

Vorab muss einem ESAWN-Knoten damit ein Ausschnitt des Aggregationsbaums bekannt sein, nämlich alle Knoten um in herum im Abstand von bis zu $k+1$ Schritten, mit denen er auch kommuniziert. Das sind $((\sum_{h=0}^{k+1} \delta^h) + k)$ Adressen und kryptographische Schlüssel.

$(\sum_{h=0}^{k} \delta^h)$ lässt sich unter der Annahme $\delta > 1$ zu $\frac{\delta^2 \cdot \delta^k - 1}{\delta - 1}$ vereinfachen. Der Speicherverbrauch liegt somit in $O(1)$ bezogen auf die Netzwerkgröße. Er hängt vom Verzweigungsgrad des Aggregationsbaums und dem ESAWN-Parameter k ab, also $O(\delta^k)$.

5.3.4 ESAWN-2

Ein ESAWN-2-Knoten muss neben der normalen Aggregation auch an Mehrheitsentscheiden seiner Kindknoten teilnehmen. Dies bedeutet, dass er Werte von seinen bis zu $2\bar{k}+1$ Schritten entfernten Kindern empfangen und speichern muss. Insgesamt also $(\sum_{h=0}^{2\bar{k}+1} \delta^h)$ Werte.

Vorab muss einem ESAWN-2-Knoten dazu ein Ausschnitt des Aggregationsbaums bekannt sein, nämlich alle Knoten um in herum im Abstand von bis zu $2\bar{k} + 1$ Schritten, mit denen er auch kommuniziert. Das sind $((\sum_{h=0}^{2\bar{k}+1} \delta^h) + \bar{k})$ Adressen und kryptographische Schlüssel. $(\sum_{h=0}^{2\bar{k}+1} \delta^h)$ lässt sich unter der Annahme $\delta > 1$ zu $\frac{\delta^2 \cdot \delta^{2\bar{k}} - 1}{\delta - 1}$ vereinfachen. Der Speicherverbrauch liegt somit in $O(1)$ bezogen auf die Netzwerkgröße. Er hängt vom Verzweigungsgrad des Aggregationsbaums und dem ESAWN-Parameter $\bar{k}$ ab, also $O(\delta^{\bar{k}})$.

5.3.5 ESAWN-NR

Bei ESAWN-NR muss ein Knoten zusätzlich zu den Informationen bei ESAWN-2 noch die versendeten Chiffrate der anderen beteiligten Knoten und von jedem Mehrheitsentscheid an dem er teilnimmt, speichern. Da ein Knoten an bis zu

$((\sum_{h=0}^{2\bar{k}+1} \delta^h) + \bar{k})$ Mehrheitsentscheiden teilnimmt, muss er $((\sum_{h=0}^{2\bar{k}+1} \delta^h) + \bar{k}) \cdot (2\bar{k})$ Chiffrate speichern.

Da dieser Mehraufwand jedoch konstant ist, liegt auch hier der Speicheraufwand in $O(1)$ bezogen auf die Netzwerkgröße. Er hängt vom Verzweigungsgrad des Aggregationsbaums und dem ESAWN-Parameter $\bar{k}$ ab, also $O(\delta^{\bar{k}})$.

5.3.6 Bewertung des Speicherverbrauchs

Schon bei den beiden Vergleichsprotokollen ist gut zu erkennen, dass AGGN mehr – wenn auch nicht große – Anforderungen an den Speicher stellt als NOAG. Einsparungen, die in der Kommunikation erzielt werden, gehen zu Lasten des Speicherverbrauchs.

So auch bei den ESAWN-Protokollen. Durch die entsprechende Leistungsfähigkeit erfordern sie wesentlich mehr Speicher, als bei den Vergleichsprotokollen erforderlich. Allerdings hängt der Speicherverbrauch nie von der Netzwerkgröße n ab. Daher ist zu erwarten, dass das Protokoll auch bei großen Netzwerken hinsichtlich des Speicherverbrauchs gut skaliert.

In Systemen, in denen selbst die genannten, recht niedrigen, Speicheranforderungen zu hoch sind, lässt sich der Speicheraufwand dennoch optimieren, wie im Anhang C am Beispiel von ESAWN verdeutlicht werden soll. Die Optimierungen verbessern jedoch nicht die Komplexitätsklasse des Speicherverbrauchs.

5.4 Sicherheit

Neben dem Energie- und Speicherverbrauch ist Sicherheit das dritte wichtige Kriterium, das im Nutzerinteresse steht. So ist zwar nun klar, welchen Arbeitsspeicher die ESAWN-Protokolle belegen und mit welchem Energieverbrauch zu rechnen ist. Offen ist jedoch bisher die Frage geblieben, welches Vertrauen der Nutzer in das am Ende eines Messvorgangs gelieferte Aggregat legen kann. Das heißt, ob das Aggregat authentisch ist und tatsächlich dem aggregierten Wert der einzelnen Messwerte entspricht.

Hierzu eignet sich die „Wahrscheinlichkeit korrekter Aggregation“, die in Abschnitt 2.2 eingeführt wurde. Sie gibt die Wahrscheinlichkeit an, mit der ein Nutzer ein authentisches Aggregat erhält.

Die mit ESAWN und bestimmten Parametern k und p erreichbare Wahrscheinlichkeit ein authentisches Aggregat zu erhalten, wird mit WKA bezeichnet. Die von ESAWN-2 und ESAWN-NR im gleichen Netzwerk realisierte Wahrscheinlichkeit dagegen mit WKA2. Diese Unterscheidung ist sinnvoll, da der bei ESAWN zum Einsatz kommenden Protokollparameter k nicht mit $\bar{k}$ gleich zu setzen ist, siehe Abschnitt 3.1.1.

WKA und WKA2 sind bei $k = \bar{k}$ nicht gleich. Es ist wahrscheinlicher, dass in einem Aggregationspfad in der Menge der $2\bar{k} + 1$ hintereinanderliegenden Knoten mehr als $\bar{k}$ Knoten korrumpiert sind, als dass mehr als k hintereinanderliegende Knoten korrumpiert sind. Es ist bei $k = \bar{k}$ daher zu erwarten, dass die WKA2 geringer ausfällt als die WKA.

5.4.1 Wahrscheinlichkeit korrekter Aggregation

k, $\bar{k}$ und p haben unterschiedlichen Einfluss auf die Wahrscheinlichkeit korrekter Aggregation (WKA):

- k verhindert bei ESAWN, durch den Einsatz von k Zeugen, einen erfolgreichen Angriff von bis zu k auf einem Aggregationspfad hintereinanderliegenden korrumpierten Knoten. Dies hat ein Beweis in [6] ergeben. Gilt dies und außerdem $p = 1$ ist ein erfolgreicher Angriff unmöglich und es gilt WKA = 100%. Allgemein gilt, dass ein höheres k zu einem höheren Sicherheitsniveau führt.
- $\bar{k}$ verhindert bei ESAWN-2 und ESAWN-NR, durch entsprechende Mehrheitsentscheide aus $2\bar{k}+1$ Knoten, einen erfolgreichen Angriff von bis zu $\bar{k}$ korrumpierten Knoten in einer Menge von $2\bar{k}+1$ hintereinanderliegenden Knoten. Dies ergibt sich aus dem Beweis in Abschnitt 3.1.4. Auch hier gilt zusammen mit $p = 1$, dass so kein erfolgreicher Angriff möglich ist und deshalb WKA2 = 100% gilt. Genauso wie k sorgt auch ein höheres $\bar{k}$ zu einem höheren Sicherheitsniveau.
- p regelt die Wahrscheinlichkeit der Überprüfung eins Aggregats durch Zeugen (bei ESAWN) oder durch Mehrheitsentscheid (bei ESAWN-2 und ESAWN-NR). Je geringer p, desto leichter ist es für den Angreifer, einen erfolgreichen Angriff durchzuführen. Der genau Einfluss von p lässt sich formal ausdrücken und wird in folgendem Absatz, 5.4.1.1, erläutert.

5.4.1.1 Einfluss von p auf die WKA(2)

Geht man davon aus, dass k bzw. $\bar{k}$ hinreichend groß gewählt sind und dem Angreifer keine Möglichkeit eines erfolgreichen Angriffs ermöglichen, dann kann der Einfluss von p auf die WKA(2) formal als

$$\text{WKA} = (1 - (\beta \cdot (1 - p)))^{\sum_{i=1}^{h-1} \delta^i} \qquad \text{aus [6]}$$

bzw.

$$\text{WKA2} = (1 - (\beta \cdot (1 - p)))^{\sum_{i=2\bar{k}+1}^{h-1} \delta^i}$$

ausgedrückt werden.

Den Formeln liegt dabei folgende Überlegung zu Grunde: Grundsätzlich muss ein korrumpierter Knoten geprüft werden, damit kein erfolgreicher Angriff möglich ist. Über die Gegenwahrscheinlichkeit ausgedrückt: Man möchte nicht, dass ein korrumpierter Knoten (Wahrscheinlichkeit β) nicht geprüft wird (Wahrscheinlichkeit $1-p$). Für einen einzelnen Knoten ergibt dies $(1 - (\beta \cdot (1 - p)))$.

Dies muss nun für alle inneren Knoten, die durch Zeugen geprüft werden (ESAWN) bzw. alle inneren Knoten, die durch einen Mehrheitsentscheid überprüft werden (ESAWN-2, ESAWN-NR), gelten. Deshalb wird der erste Teil der Formel noch durch die entsprechende Zahl innerer Knoten potentiert, um die Gesamtwahrscheinlichkeit zu erhalten.

Bei ESAWN sind dies alle innere Knoten, da jeder durch bis zu k Zeugen überprüft wird. Bei ESAWN-2 und ESAWN-NR sind dies alle Knoten, die durch einen Mehrheitsentscheid geprüft werden. Also alle, die mindestens $2\bar{k}+1$ Schritte von der Senke

entfernt sind und für die damit genügend Knoten für einen Mehrheitsentscheid zur Verfügung stehen.

Die Diagramme in den Abbildungen 5.8 und 5.9 vergleichen die Simulationsläufen der GloMoSim- und SecuSim-Implementierungen mit der theoretisch hergeleiteten Formel.

Die Wahl der Systemparameter k und $\bar{k}$ erwies sich dabei als kritisch, da diese zum einen hinreichend groß gewählt sein sollten, andererseits aber nicht zu groß sein durften. Bei zu großem $\bar{k}$ werden keine Mehrheitsentscheide mehr durchgeführt, und die WKA2 ist unabhängig von p immer 1, bei entsprechend hohen Kosten. Die Wahl fiel daher auf $k = \bar{k} = \lceil h \cdot \beta \rceil$. Dies resultiert aus der Idee, dass auf einem Aggregationspfad der Länge h etwa $h \cdot \beta$ Angreifer zu erwarten sind. Die Netzwerkstruktur wurde mit $n = 1000$ Knoten und $\delta = 2$ konstant gehalten.

Betrachtet man zuerst die beiden oberen Diagramme, die aus der formalen Berechnung entstanden sind, fallen sofort die recht schnell abfallenden Kurven und damit eine schnell abnehmende WKA bzw. WKA2 auf. Der Effekt ist umso stärker, je größer die Korrumpierungswahrscheinlichkeit β gewählt ist. Dabei sind die Verläufe der WKA2 und WKA fast identisch, da sich die Zahl der relevanten inneren Knoten mit der potenziert wird, nur minimal unterscheiden.

Die beiden unteren Diagramme, welche die simulierten Ergebnisse mit gleichen Netzparametern darstellen, scheinen auf den ersten Blick dem formalen Ansatz zu widersprechen, zeigen sie doch eine deutliche niedrigere WKA und WKA2. Die Diskrepanz ist dabei umso größer, je größer β gewählt wurde.

Der Unterschied erklärt sich, wenn man sich noch einmal die Definition einer korrekten Aggregation vor Augen führt:

Eine Aggregation ist nur dann erfolgreich, wenn sowohl jeder korrumpierte Aggregationsknoten geprüft wurde *und* außerdem nie mehr als k Knoten hintereinander bzw. $\bar{k}$ in $2\bar{k} + 1$ hintereinanderliegenden Aggregationsknoten korrumpiert wurden.

Die beiden Parameter k und $\bar{k}$ wurden mit $\beta \cdot h$ so groß gewählt, dass sie der mittleren erwarteten Angreiferzahl pro Aggregationspfad entsprechen. Zwei Faktoren sorgen jedoch dafür, dass es im Simulationsergebnis trotzdem immer wieder Fälle gibt, bei denen alleine die „ungünstige“ Platzierung der korrumpierten Knoten für ein Scheitern der Aggregation sorgt:

1. Die simulierten Aggregationsbäume sind nicht gleichmäßig. Sie erfüllen die gegebenen Netzparameter, wie Netzgröße n. Der Verzweigungsgrad δ wird beispielsweise im Mittel erfüllt, es gibt jedoch Knoten mit mehr und Knoten mit weniger Kindknoten. Daraus resultieren Aggregationsbäume mit kürzeren und auch deutlich längeren Aggregationspfaden, als der im theoretischen Modell berücksichtigen Länge. Auf längeren Aggregationspfaden ist die Wahrscheinlichkeit einer größeren Angreiferzahl auch höher.

2. Die Menge der korrumpierten Aggregationspfade wird zufällig ausgewählt. So kann es passieren, dass die korrumpierten Knoten in einem Fall auf hinreichend viele unterschiedliche Aggregationspfade verteilt sind und so unter der kritischen Marke von k bzw. $\bar{k}$ in $2\bar{k} + 1$ Knoten hintereinanderliegenden Knoten bleiben. Es gibt jedoch genauso auch Fälle, in denen die Knoten so korrumpiert werden, dass die Aggregation selbst bei $p = 1$ scheitert.

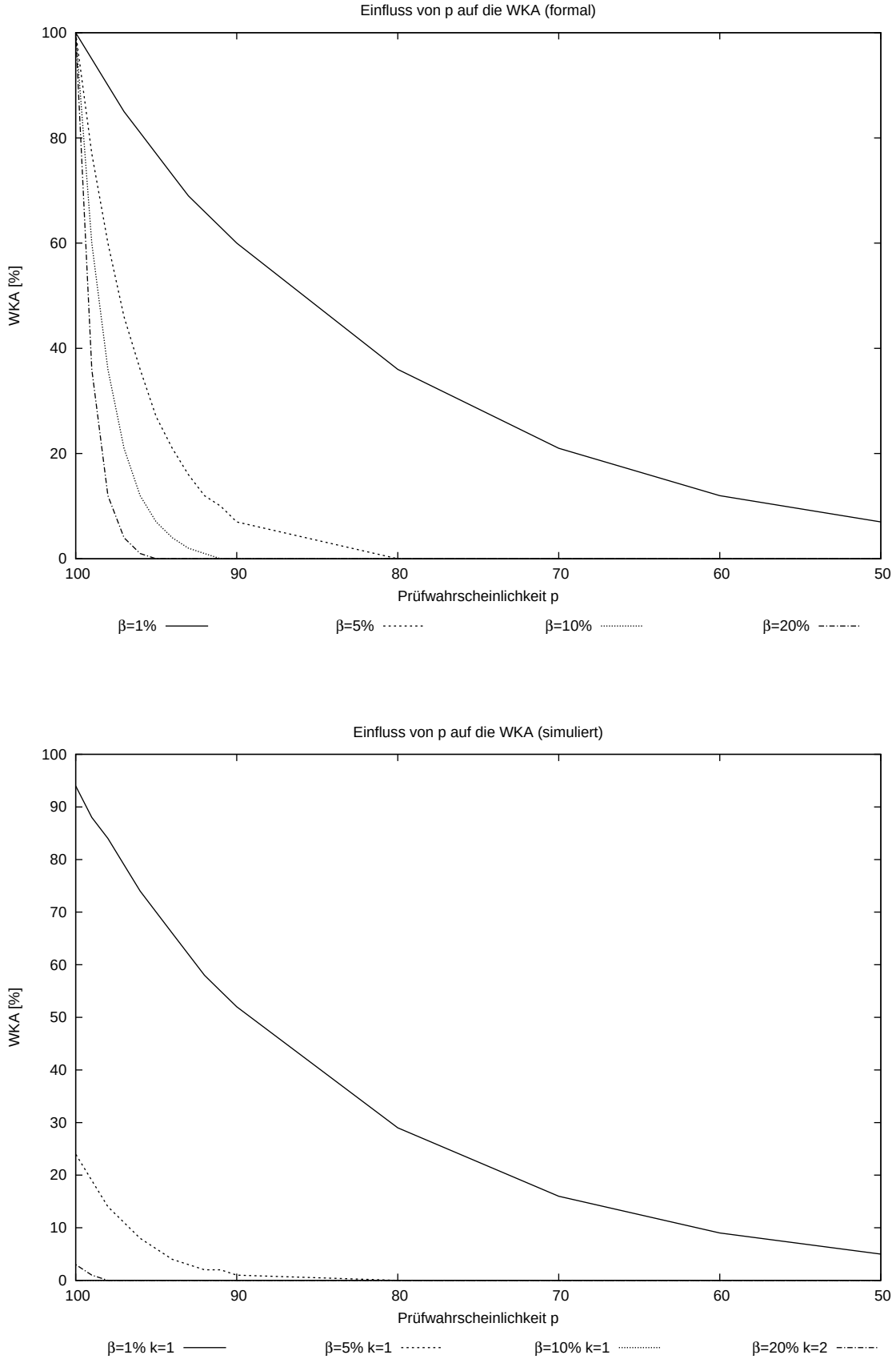


Abbildung 5.8: WKA formal und simuliert im Vergleich, k hinreichend groß

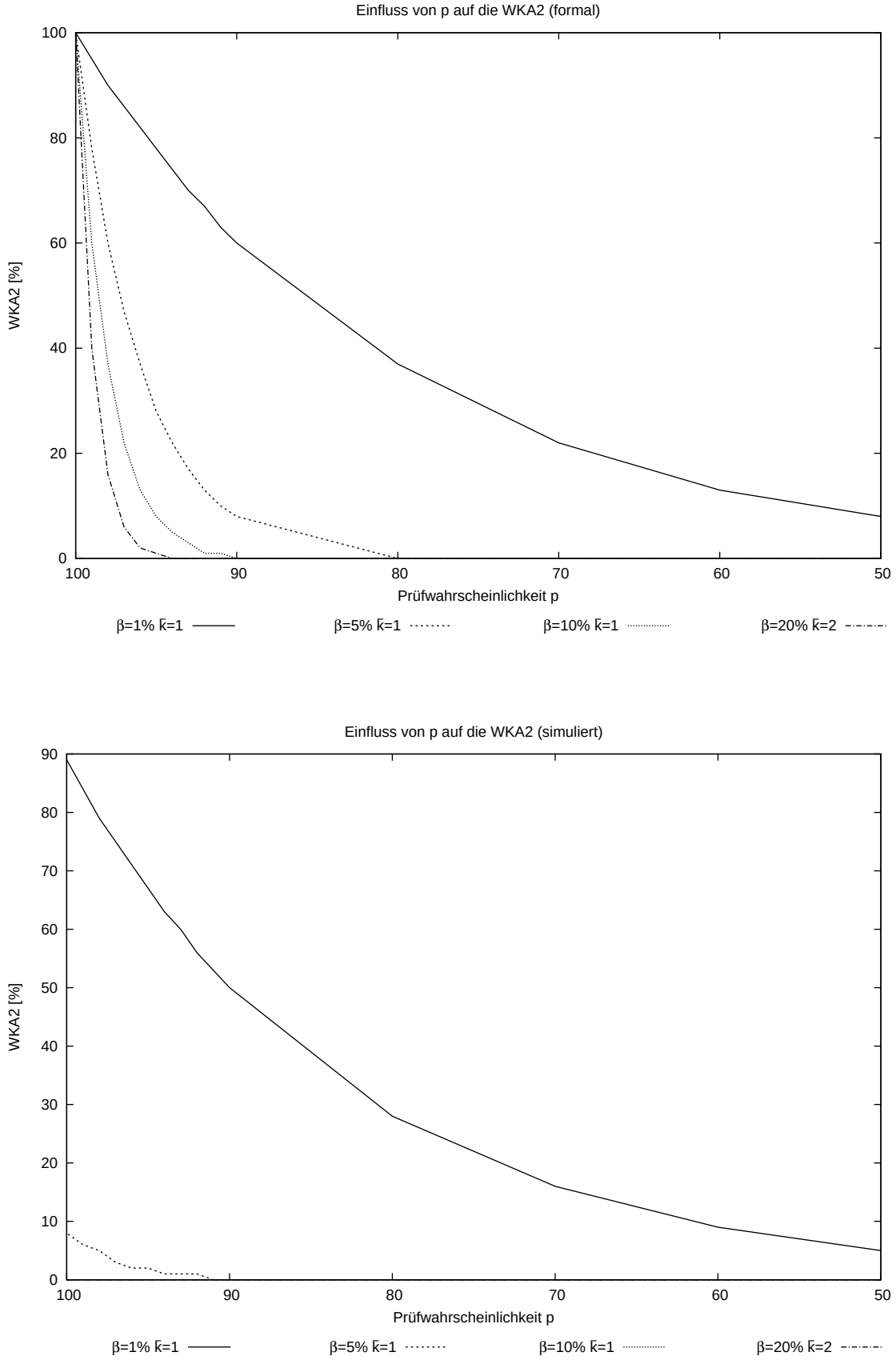


Abbildung 5.9: WKA2 formal und simuliert im Vergleich, $\bar{k}$ hinreichend groß

Der Abstand zwischen den Kurven des formalen Modells und den simulierten Ergebnissen liegt also alleine an den Fällen, in denen das gewählte k und $\bar{k}$ nicht ausreichend ist, um die Aggregation erfolgreich abzuschließen. Diese Fälle sind umso wahrscheinlicher, je größer β gewählt ist, weshalb der Abstand mit zunehmendem β auch größer wird.

Dagegen hilft nur ein noch größeres k bzw $\bar{k}$. Dies führt jedoch dazu, dass sich auf den meisten Aggregationspfaden nicht mehr genügend Zeugen für einen Mehrheitsentscheid befinden. Damit kommt fast die gesamte Energieersparnis im Vergleich zu NOAG durch Paketaggregation zustande. Der eigentliche Kern der ESAWN-2- und ESAWN-NR-Protokolle, die Nutzung von Mehrheitsentscheiden, wird kaum noch genutzt, da nur noch auf sehr langen Aggregationspfaden auch tatsächlich aggregiert wird.

5.4.1.2 Einfluss von $k/\bar{k}$ auf die WKA(2)

Die Annahme eines hinreichend groß gewählten Parameters k bzw. $\bar{k}$ ist allerdings wenig energieeffizient. Mit reduziertem k bzw. $\bar{k}$ steigt jedoch, wie im vorherigen Abschnitt erklärt, die Wahrscheinlichkeit, dass sich korrumpierte Knoten in einer für das gewählte ESAWN-Protokoll ungünstigen Platzierung befinden. Diese Tatsache macht sich durch eine reduzierte WKA bemerkbar.

Die Probleme, die bei der isolierten Betrachtung von p auftraten, gibt es bei der Betrachtung des Einflusses von $k/\bar{k}$ nicht, da hier $p = 1$ gewählt werden kann. Dadurch wird ein Scheitern der Aggregation durch eine nicht erfolgte Überprüfung unterbunden.

Die nächsten Simulationsergebnisse, dargestellt in den Abbildungen 5.10 bis 5.13, geben im Detail darüber Aufschluss, bei welcher Wahl der Parameter k und $\bar{k}$ mit welcher WKA(2) zu rechnen ist. Dazu werden Simulationen mit ESAWN und ESAWN-2 verglichen, da ESAWN-NR und ESAWN-2 die gleiche WKA liefern und sich nur in den Kommunikationswegen unterscheiden.

Dabei ist deutlich zu erkennen, dass bei $k = \bar{k}$ die WKA2 niedriger ausfällt als die WKA. Dies ist auch der Grund, weshalb bereits im Abschnitt 3.1.1 der Protokollparameter k für ESAWN-2 und ESAWN-NR in $\bar{k}$ umbenannt wurde. Tatsächlich liefern die beiden Parameter unterschiedliche Sicherheitsgarantien. Um eine bestimmte von ESAWN her bekannte WKA zu erreichen, reicht es also *nicht* aus, einfach $k = \bar{k}$ zu setzen. Dies liegt daran, dass es bei gleichverteilten korrumpierten Knoten wahrscheinlicher ist, dass in $2\bar{k}+1$ hintereinanderliegenden Knoten mindestens $\bar{k}$ *beliebige* Knoten korrumpiert sind, als dass mindestens k *direkt* hintereinanderliegende Knoten korrumpiert sind.

Den größten Einfluss auf die WKA(2) hat dabei der Anteil korrumpierter Knoten β. Ein größeres β sorgt für eine geringere WKA(2). Genau umgekehrt verhält es sich erwartungsgemäß mit den Protokollparametern k und $\bar{k}$, je größer diese sind, desto wahrscheinlicher ist eine korrekte Aggregation.

Außerdem nimmt die WKA(2) mit steigender Netzwerkgröße ab. Durch das größere Netzwerk kommt es zu längeren Aggregationspfaden und damit einer höheren Wahrscheinlichkeit, dass sich korrumpierte Knoten in einer für die erfolgreiche Aggregation ungünstigen Platzierung befinden. Für eine fehlerhafte Gesamtaggregation

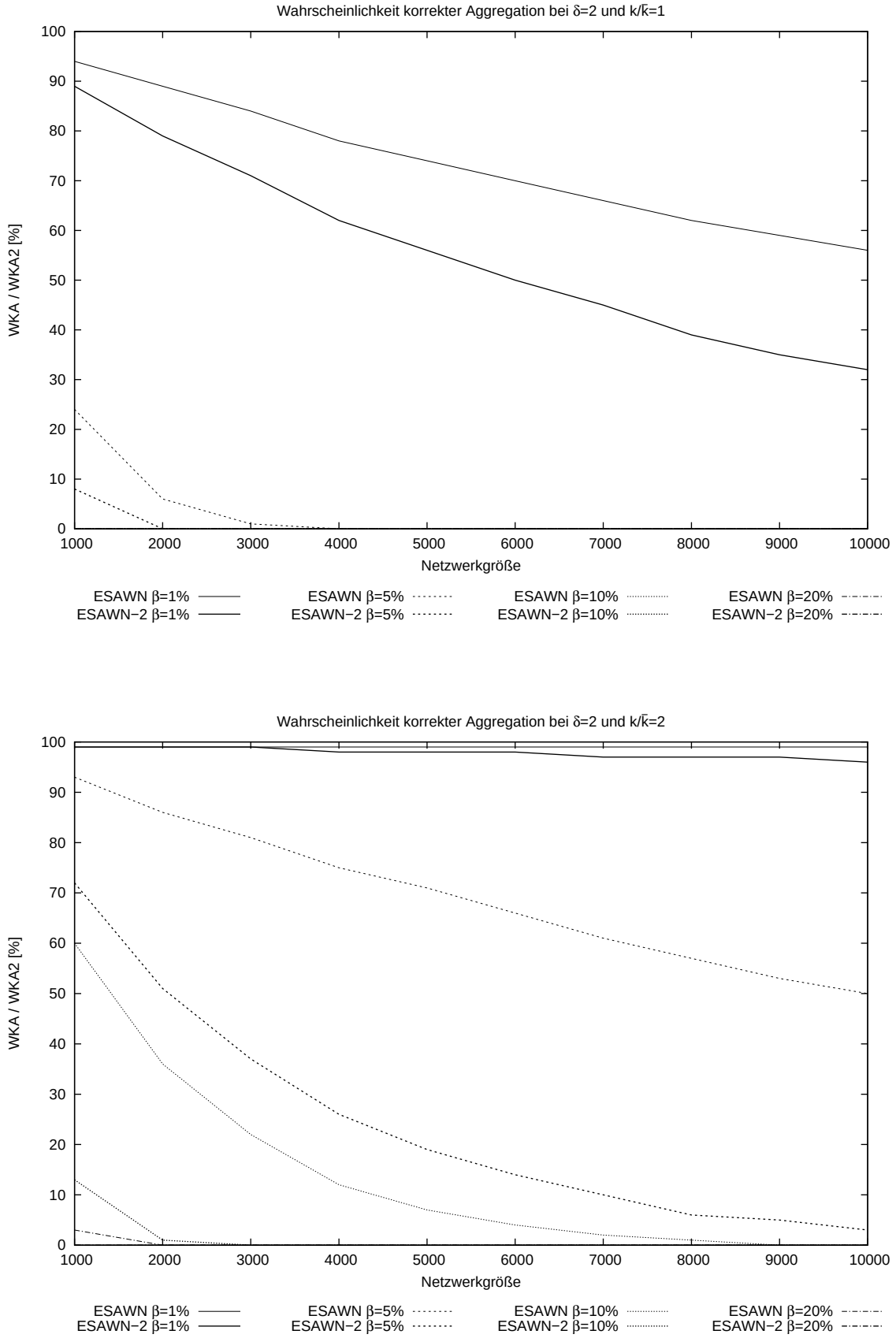


Abbildung 5.10: WKA und WKA2 mit $p = 1$, $\delta = 2$ und variablem $k/\bar{k}$ (I)

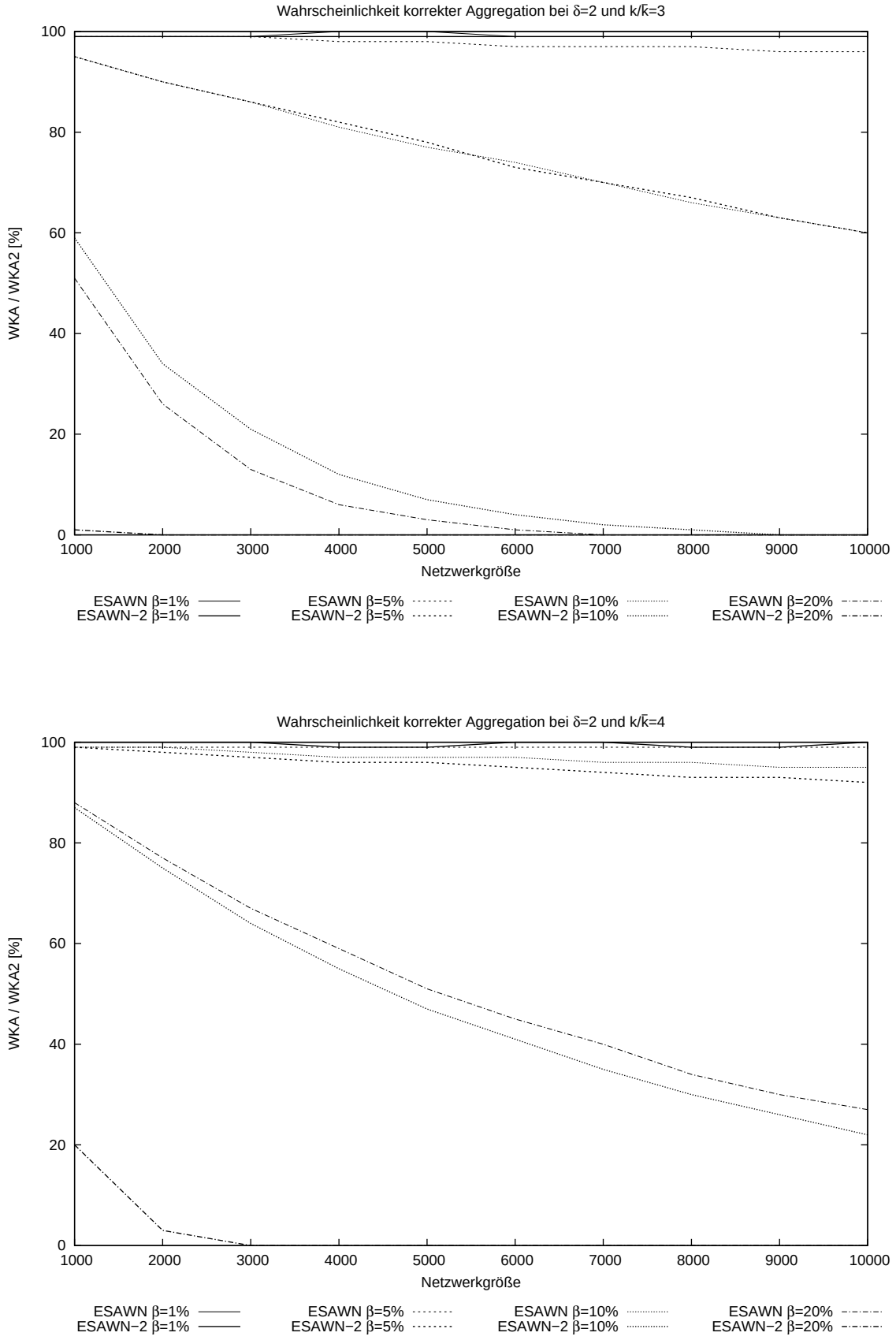


Abbildung 5.11: WKA und WKA2 mit $p = 1$, $\delta = 2$ und variablem $k/\bar{k}$ (II)

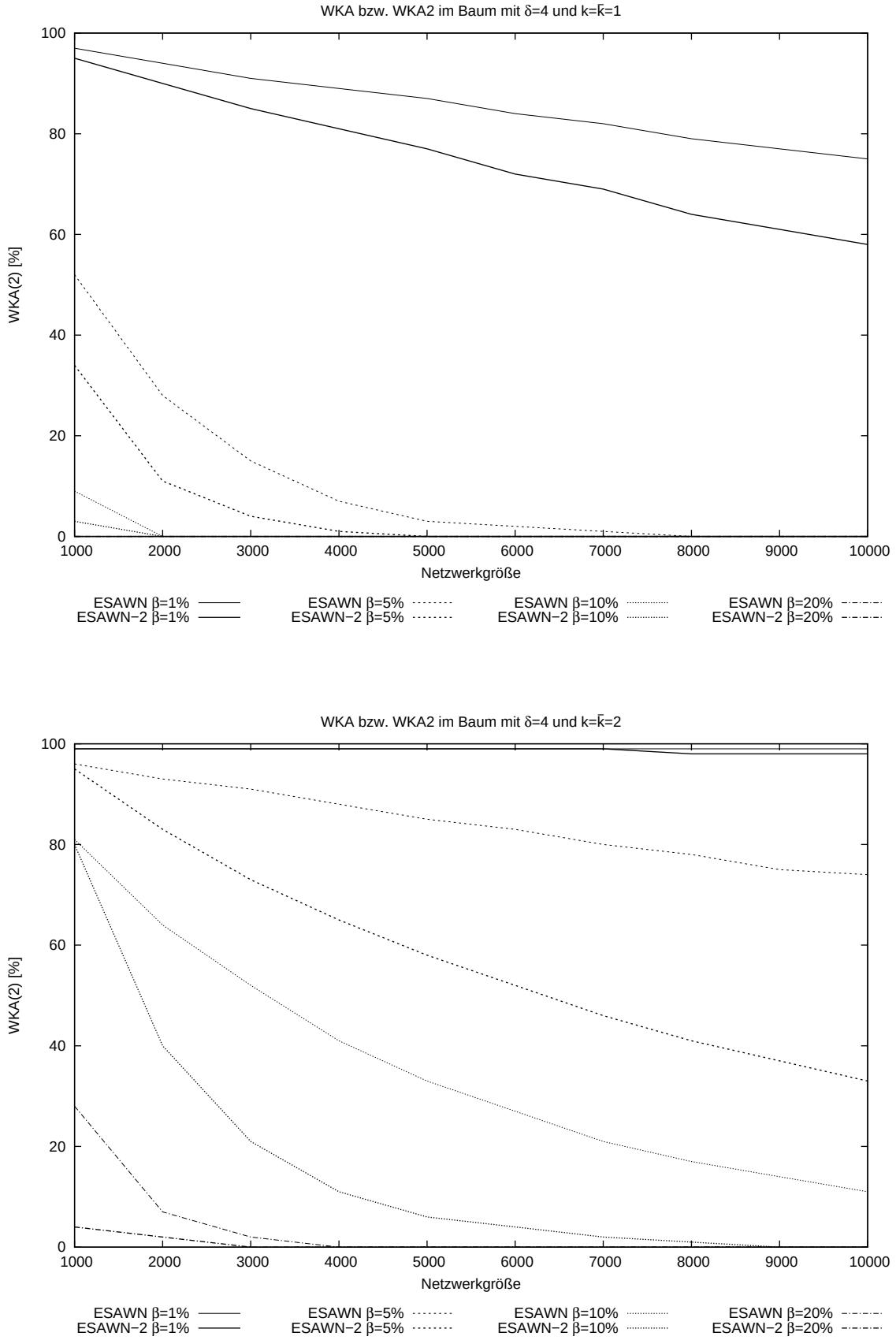


Abbildung 5.12: WKA und WKA2 mit $p = 1$, $\delta = 4$ und variablem $k/\bar{k}$ (I)

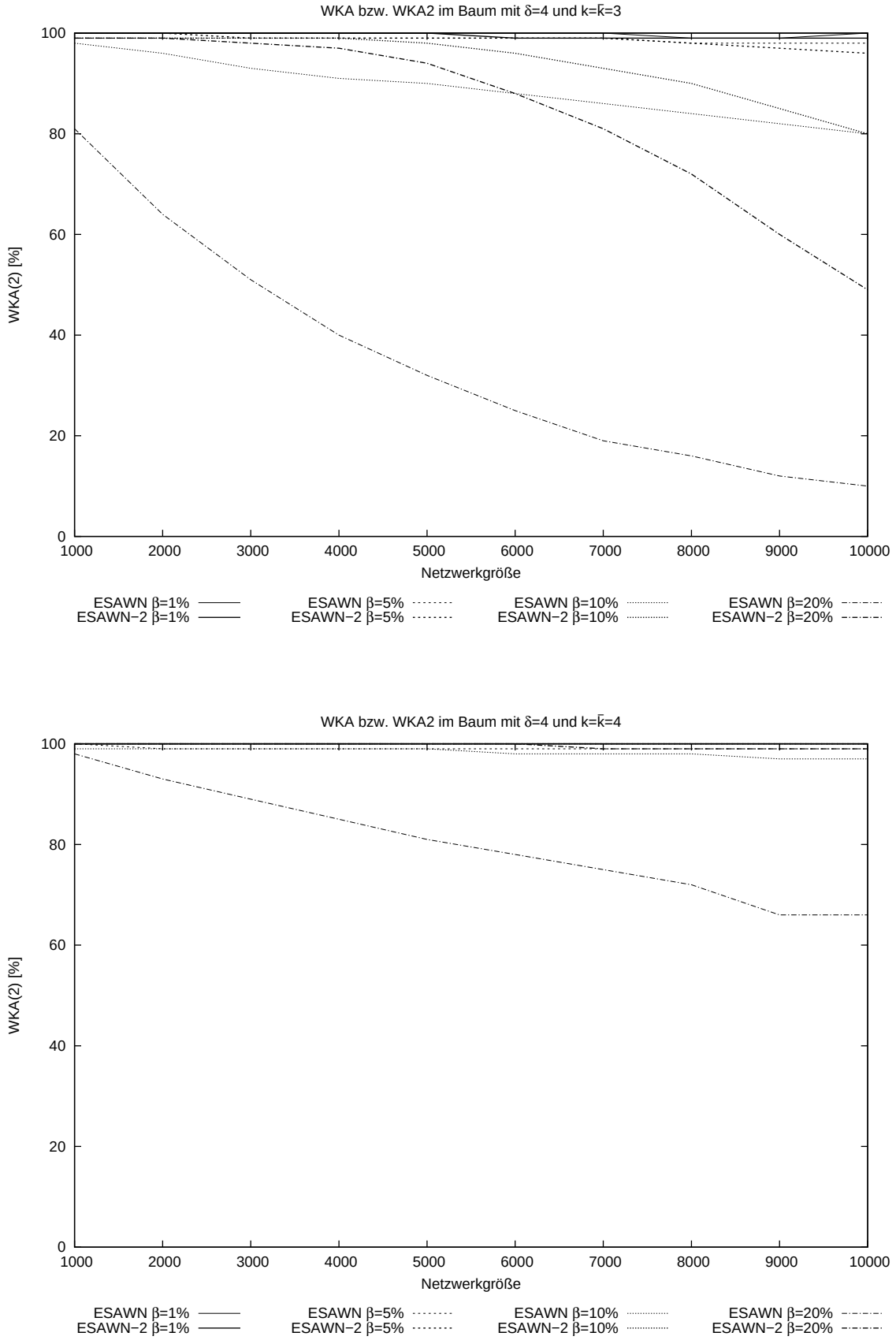


Abbildung 5.13: WKA und WKA2 mit $p = 1$, $\delta = 4$ und variablem $k/\bar{k}$ (II)

reicht es bereits aus, wenn auch nur auf einem der vielen Aggregationspfade solch eine Konstellation auftritt.

Für längere Aggregationspfade sorgt auch ein geringerer Verzweigungsgrad im Aggregationsbaum. Je kleiner δ ausfällt, desto kleiner die WKA bei ansonsten gleicher Konfiguration.

5.4.2 Mindestsicherheit

Bislang wurde der Energieverbrauch und die Wahrscheinlichkeit einer korrekten Aggregation völlig getrennt voneinander betrachtet. Jedoch wird in den seltensten Fällen ein Nutzer konkrete Werte für k, $\bar{k}$ und p parat haben, die seinem Anwendungsszenario entsprechen. Viel wahrscheinlicher ist die Vorgabe einer konkreten Mindestsicherheit und der damit verbundenen Frage nach dem dabei möglichen Einsparpotential durch die ESAWN-Protokolle.

Dabei wurde von einem konkreten Nutzer ausgegangen, der eine WKA von mindestens P Prozent wünscht. Er kennt die Netzgröße n, den Verzweigungsgrad δ und geht von einem Angreifer mit dem maximalen Potential $\mathcal{B}$ Knoten zu korrumpieren aus.

Daraus resultiert die Frage: „Wie müssen $\bar{k}$ und p gewählt werden, und mit welchem Energieverbrauch ist zu rechnen?“ Dem soll in den letzten Diagrammen dieses Kapitels nachgegangen werden:

	$\beta = 0{,}01\%$		$\beta = 0{,}1\%$	
	Diagramm	Tabelle	Diagramm	Tabelle
ESAWN	siehe [41]			
ESAWN-2	Abb. 5.15	5.1	Abb. 5.16	5.2
ESAWN-NR	Abb. 5.17	5.3	Abb. 5.18	5.4

Aus Gründen der Übersichtlichkeit, sind nicht immer alle Kurven für $P \in \{50\%, 60\%, 70\%, 80\%, 90\%, 100\%\}$ dargestellt. Die fehlenden Kurven überlagern sich dann mit anderen Kurven aufgrund der gleichen Parameterkonfiguration.

Da es immer mehrere Möglichkeiten gibt, $\bar{k}$ und p so zu wählen, dass die WKA $\geq P$ ist, wurde im Folgenden immer die Kombination gewählt, die sich energetisch am günstigsten verhält.

Eine vergleichbare Analyse des Protokolls ESAWN findet sich in [41].

5.4.2.1 ESAWN-2

Das Diagramm in Abbildung 5.15 stellt die Situation mit ESAWN-2 dar, bei $\beta = 1\%$. Auffällig sind die Sprünge, die auftreten, wenn die bisherige Wahl von $\bar{k}$ selbst bei $p = 1$ nicht mehr ausreicht, um die geforderte Mindestwahrscheinlichkeit zu erreichen. Da $\bar{k}$ nur in diskreten Schritten gewählt werden kann, bringt der Wechsel auf ein höheres $\bar{k}$ einen Sprung der Energiekosten mit sich.

Dies ist beispielsweise bei $P = 80\%$ und dem Sprung von $n = 4000$ auf $n = 5000$ Knoten der Fall. Von dem bisher genutzten $\bar{k} = 1$ muss auf $\bar{k} = 2$ gewechselt werden.

Interessant dabei ist, dass sich in diesem Fall bei $\bar{k} = 2$ gleich auch $p = 1$ ergibt und kein geringeres p verwendet werden kann. Die Prüfwahrscheinlichkeit hat hier einen so großen Einfluss auf die WKA, dass eine Absenkung nicht mehr in Frage kommt.

Der dem Diagramm zugehörigen Tabelle 5.1 kann man entnehmen, dass eine Veränderung an p sich generell nur für kleine Netzwerke und geringe Mindestwahrscheinlichkeiten lohnt.

Bei $\beta = 0{,}1$, also der Situation, wie sie in Abbildung 5.16 dargestellt ist, ergibt sich ein völlig anderes Bild. Die Kurven liegen sehr nahe beieinander. Dies liegt daran, dass in fast allen Fällen die Konfiguration $(\bar{k} = 3, p = 1)$ bzw. $(\bar{k} = 4, p = 1)$ gewählt wurde. Eine niedrige Konfiguration kommt nicht in Frage, da selbst die geringen Ansprüche einer Mindestwahrscheinlichkeit von $P = 50\%$ nicht mehr erfüllt werden. Dies liegt an der recht hohen Wahrscheinlichkeit, dass ein Knoten korrumpiert ist, von $\beta = 0{,}1$.

Die Kurven von $(\bar{k} = 3, p = 1)$ und $(\bar{k} = 4, p = 1)$ liegen dabei sehr nahe beieinander. Dies liegt an dem selben Grund, wie er bereits bei der Betrachtung in Abschnitt 5.2.2 diskutiert wurde, den für einen Mehrheitsentscheid zu kurzen Aggregationspfaden, bei denen ESAWN-2 in das direkte Versenden des Messwertes an die Senke wechselt.

5.4.2.2 ESAWN-NR

Die letzten beiden Abbildungen zeigen die gleiche Situation für das ESAWN-NR Protokoll. Da ESAWN-2 und ESAWN-NR bei gleicher Konfiguration die gleiche WKA2 liefern, sollte man annehmen können, dass hier die gleichen Konfigurationstabellen zum Einsatz kommen wie bei ESAWN-2. Ein Blick auf die Tabellen 5.3 und 5.4 zeigt jedoch, dass dies nicht der Fall ist.

Der Grund liegt darin begründet, dass für eine gegebene Mindestwahrscheinlichkeit P immer die günstigste Konfiguration gewählt werden sollte. Bei ESAWN-NR ergibt sich nun die Situation, dass stellenweise eine Konfiguration mit höherem $\bar{k}$ energetisch günstiger ist, als die ansonsten gleiche Konfiguration mit kleinerem $\bar{k}$.

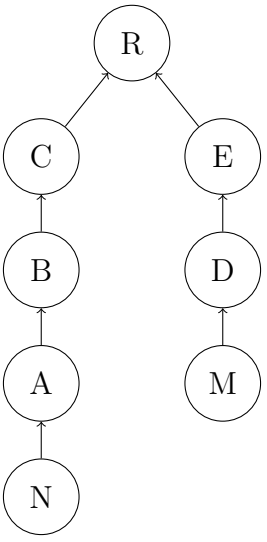


Abbildung 5.14: Einfluss der Pfadlänge auf Mehrheitsentscheide

An einem beispielhaft gewählten Aggregationsbaum 5.14 lässt sich dieser Umstand illustrieren. Zuerst betrachtet man den Aggregationspfad von Blattknoten N, auf dem die Länge grade ausreicht, dass ESAWN-NR mit bei $\bar{k} = 1$ Mehrheitsentscheide durchführen kann. Der zweite Pfad in diesem Aggregationsbaum erlaubt es ESAWN-NR bei gleichem $\bar{k}$ keinen Mehrheitsentscheid mehr durchzuführen. Im zweiten Fall wird ESAWN-NR daher die Messwerte der Blattknoten direkt an die Senke schicken, was – abhängig von der genauen Netzwerkkonfiguration – günstiger sein kann, als der erste Fall mit Mehrheitsentscheiden.

Da die Sensorknoten mit ESAWN-NR diesen Umstand nicht erfassen können, kommt es zu dieser, auf den ersten Blick paradoxen, Situation.

Während bei $\beta = 0{,}01$ noch Unterschiede bei verschieden gewählten Mindestwahrscheinlichkeiten ersichtlich sind, ist dies bei $\beta = 0{,}1$ nicht mehr der Fall. Es wird generell $(\bar{k} = 4, p = 1)$ gewählt, weil dies jeweils die energetisch günstigste Möglichkeit ist. In dieser Konfiguration kommt es hier jedoch zu keinerlei Aggregationen oder Mehrheitsentscheiden mehr, die Ersparnis im Vergleich zu NOAG ergibt sich lediglich aus der Paketaggregation der Daten.

In den untersuchten Netzwerkgrößen und Konfigurationen ist ESAWN-NR somit nur bei recht kleiner Korrumpierungswahrscheinlichkeit β sinnvoll einsetzbar.

Aus den Tabellen ist außerdem ersichtlich, dass der Einsatz eines $p < 1$ nur in sehr kleinen Netzwerken oder bei kleinem P in Frage kommt. Dies deckt sich mit den Erkenntnissen aus der formalen Betrachtung von p in Abschnitt 5.4.1.1 in dem sich der starke Einfluss von β und n auf die WKA(2) herausstellte.

5.5 Zusammenfassung

ESAWN-2 und ESAWN-NR sind deutlich leistungsfähiger als ESAWN. Ein Vergleich der Energiekosten zeigt aber, dass dies seinen Preis hat. Der eingeführte Systemparameter p könnte diese Kosten zwar reduzieren, senkt aber in vielen Fällen die Wahrscheinlichkeit einer korrekten Aggregation auf ein indiskutabel niedriges Niveau. ESAWN-2 und ESAWN-NR haben ihre Berechtigung, wenn die Korrumpierungswahrscheinlichkeit der Knoten (β) niedrig liegt. Steigt β auf 0,1 oder höher, bietet ein nur paketaggregierender Datentransport jedoch ein höheres Sicherheitsniveau bei gleichen oder niedrigeren Energiekosten.

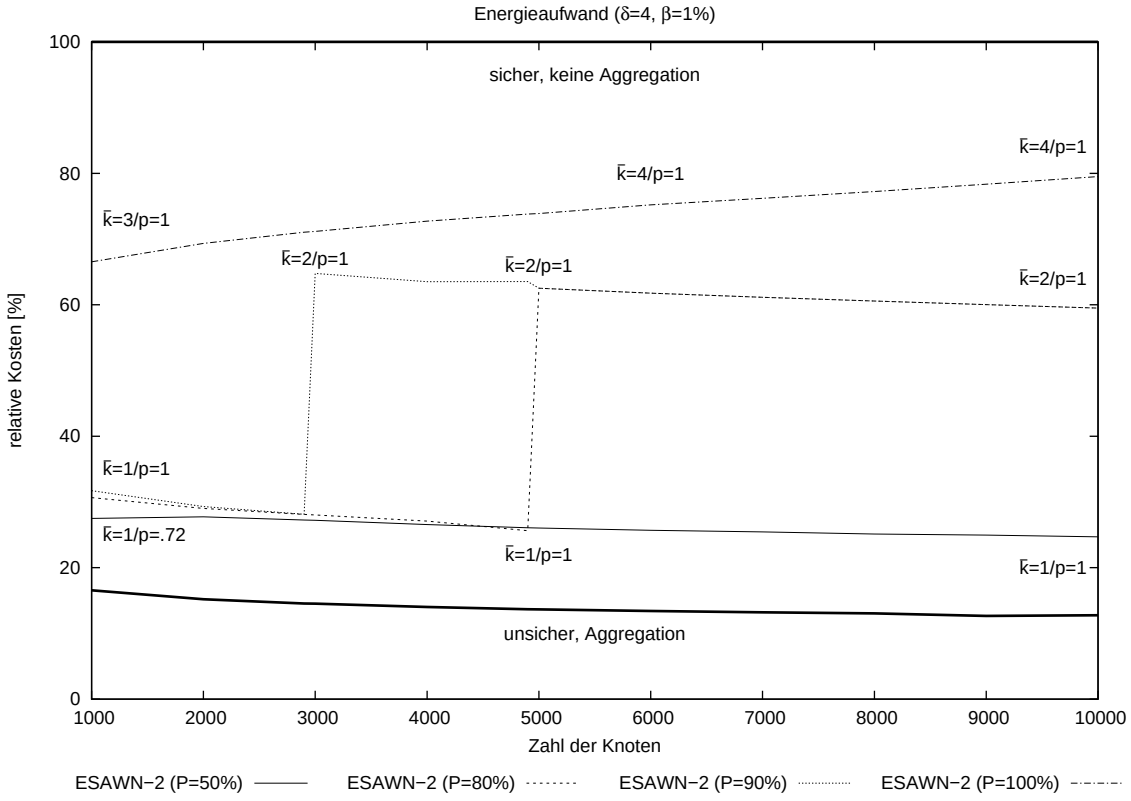


Abbildung 5.15: Energieverbrauch bei gegebener Mindestsicherheit und $\beta = 0{,}01$ in ESAWN-2

WKA2	50%		60%		70%		80%		90%		100%	
n	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$
1000	1	72	1	83	1	89	1	93	1	100	3	100
2000	1	89	1	91	1	95	1	98	1	100	3	100
3000	1	94	1	96	1	98	1	100	2	100	3	100
4000	1	96	1	98	1	99	1	100	2	100	3	100
5000	1	97	1	99	1	100	2	100	2	100	3	100
6000	1	98	1	99	1	100	2	100	2	100	4	100
7000	1	99	1	100	1	100	2	100	2	100	4	100
8000	1	99	1	100	2	100	2	100	2	100	4	100
9000	1	100	1	100	2	100	2	100	2	100	4	100
10000	1	100	1	100	2	100	2	100	2	100	4	100

Tabelle 5.1: Parameterwahl zu den Simulationsreihen aus Abbildung 5.15

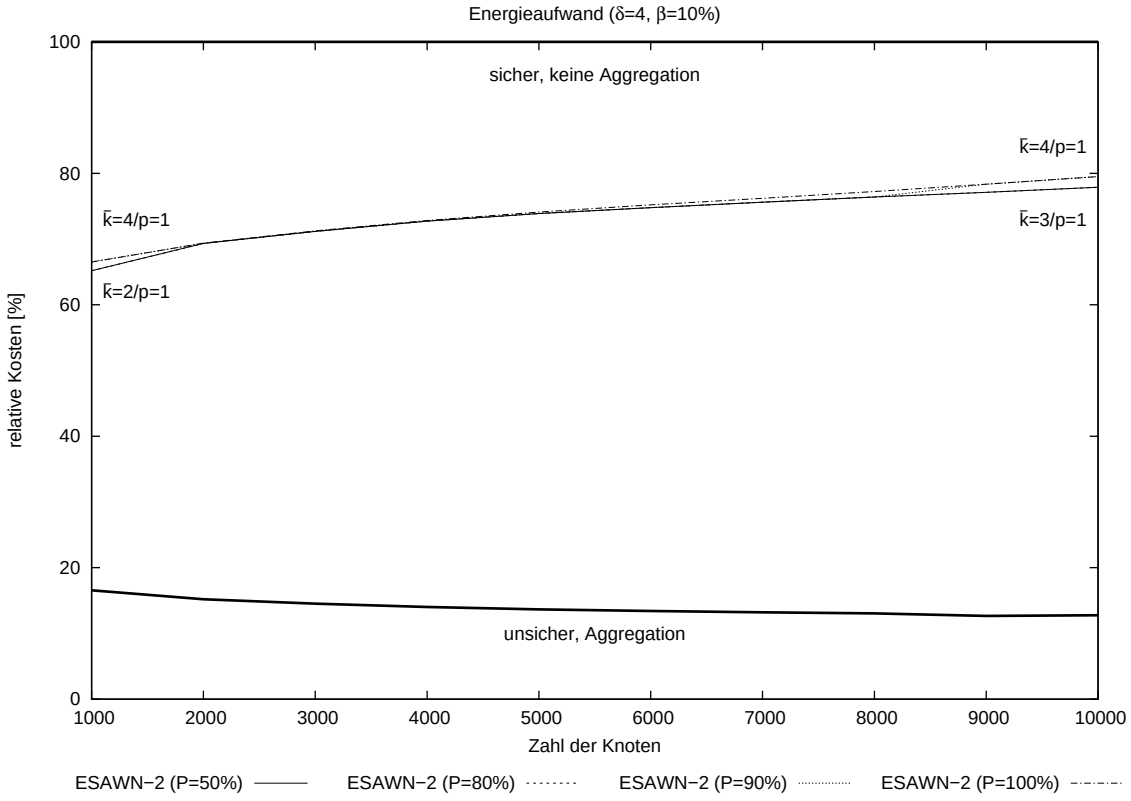


Abbildung 5.16: Energieverbrauch bei gegebener Mindestsicherheit und $\beta = 0{,}1$ in ESAWN-2

WKA2	50%		60%		70%		80%		90%		100%	
n	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$
1000	2	100	2	100	2	100	2	100	3	100	4	100
2000	3	100	3	100	3	100	3	100	3	100	4	100
3000	3	100	3	100	3	100	3	100	3	100	4	100
4000	3	100	3	100	3	100	3	100	3	100	4	100
5000	3	100	3	100	3	100	3	100	3	100	4	100
6000	3	100	3	100	3	100	3	100	3	100	4	100
7000	3	100	3	100	3	100	3	100	3	100	4	100
8000	3	100	3	100	3	100	3	100	3	100	4	100
9000	3	100	3	100	3	100	3	100	4	100	4	100
10000	3	100	3	100	3	100	3	100	4	100	4	100

Tabelle 5.2: Parameterwahl zu den Simulationsreihen aus Abbildung 5.16

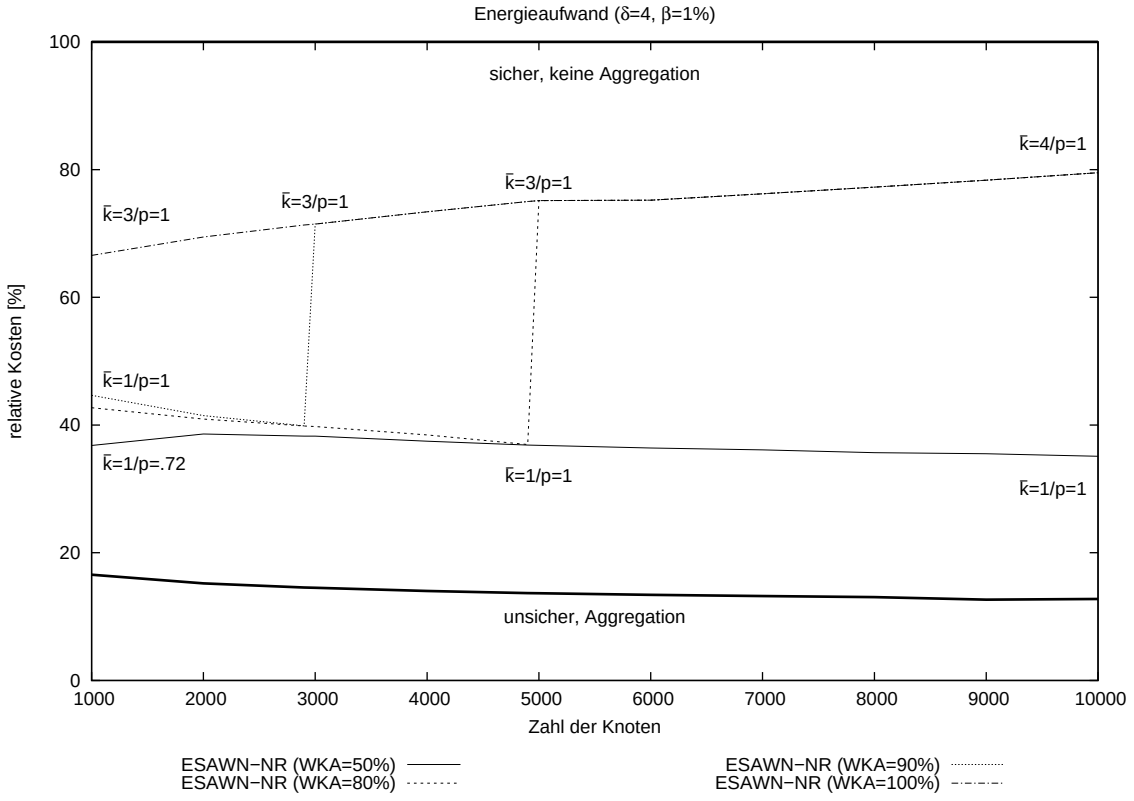


Abbildung 5.17: Energieverbrauch bei gegebener Mindestsicherheit und $\beta = 0{,}01$ in ESAWN-NR

WKA2	50%		60%		70%		80%		90%		100%	
n	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$
1000	1	72	1	83	1	89	1	93	1	100	3	100
2000	1	89	1	91	1	95	1	98	1	100	3	100
3000	1	94	1	96	1	98	1	100	3	100	3	100
4000	1	96	1	98	1	99	1	100	3	100	3	100
5000	1	97	1	99	1	100	3	100	3	100	3	100
6000	1	98	1	99	1	100	4	100	4	100	4	100
7000	1	99	1	100	1	100	4	100	4	100	4	100
8000	1	99	1	100	2	100	4	100	4	100	4	100
9000	1	100	1	100	2	100	4	100	4	100	4	100
10000	1	100	1	100	2	100	4	100	4	100	4	100

Tabelle 5.3: Parameterwahl zu den Simulationsreihen aus Abbildung 5.17

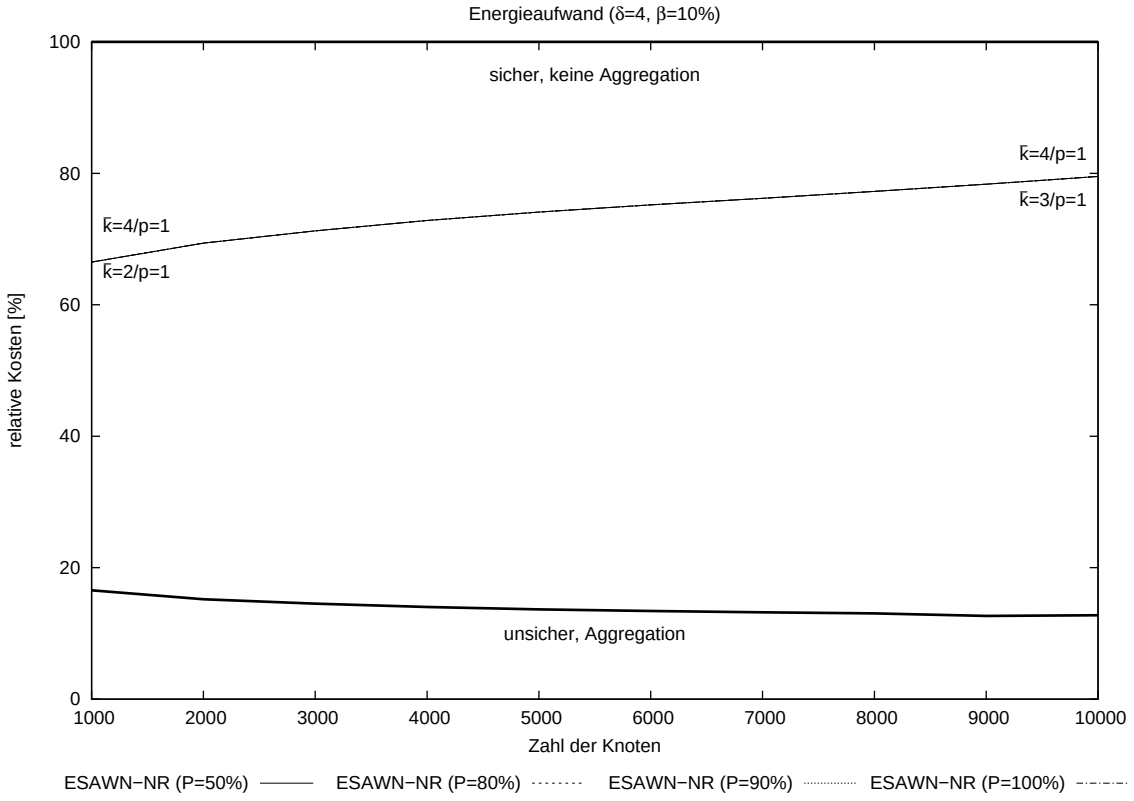


Abbildung 5.18: Energieverbrauch bei gegebener Mindestsicherheit und $\beta = 0{,}10$ in ESAWN-NR

WKA2	50%		60%		70%		80%		90%		100%	
n	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$	$\bar{k}$	$p \cdot 100$
1000	4	100	4	100	4	100	4	100	4	100	4	100
2000	4	100	4	100	4	100	4	100	4	100	4	100
3000	4	100	4	100	4	100	4	100	4	100	4	100
4000	4	100	4	100	4	100	4	100	4	100	4	100
5000	4	100	4	100	4	100	4	100	4	100	4	100
6000	4	100	4	100	4	100	4	100	4	100	4	100
7000	4	100	4	100	4	100	4	100	4	100	4	100
8000	4	100	4	100	4	100	4	100	4	100	4	100
9000	4	100	4	100	4	100	4	100	4	100	4	100
10000	4	100	4	100	4	100	4	100	4	100	4	100

Tabelle 5.4: Parameterwahl zu den Simulationsreihen aus Abbildung 5.18

6. Zusammenfassung und Ausblick

Ziel dieser Diplomarbeit war es, ein Verfahren zur authentischen Datenaggregation in drahtlosen Sensornetzen zu entwerfen. Auf Basis des ESAWN-Protokoll galt es, die Nachteile des in Vorarbeiten entwickelten ESAWN zu analysieren und zu beseitigen. Ergebnis ist das hier vorgestellte Protokoll ESAWN-2. Es ist funktional dem ESAWN-Protokoll überlegen und zeigt sich bei Störungen durch korrumpierte Knoten robuster: Ein laufender Aggregationsvorgang wird durch korrumpierte Aggregationsknoten nicht gestört. Falsche Alarmmeldungen, wie sie bei ESAWN möglich waren, sind mit ESAWN-2 ausgeschlossen.

In einem zweiten Schritt wurde das Protokoll um einen Mechanismus ergänzt, der Non-Repudiation bietet. Das daraus resultierende Protokoll ESAWN-NR bietet die Möglichkeit andere legitime Knoten eines Sensornetz über einen als korrumpiert erkannten Aggregationsknoten zu informieren und somit Schritte für seinen Ausschluss aus dem Netzwerk einzuleiten. Durch die mit ESAWN-NR realisierte Non-Repudiation kann ein Knoten ein abgesendetes Aggregat nicht mehr abstreiten. Einem Dritten, beispielsweise den anderen Knoten des Aggregationspfades, kann dadurch bewiesen werden, dass der Knoten Urheber eines korrumpierten Aggregats ist.

Beide Verfahren arbeiten, ähnlich wie bereits ESAWN, nur bis zu einer gewissen Anzahl an korrumpierten Aggregationsknoten fehlerfrei. Darüberhinaus steigt die Wahrscheinlichkeit kein authentisches Aggregat mehr zu erhalten. In Abhängigkeit der zu erwartenden Angreiferstärke lässt sich über die Protokollparameter $\bar{k}$ und p der Tradeoff zwischen erzielter Authentizität und benötigter Energie szenariobezogen einstellen. ESAWN-2 und ESAWN-NR zeigen sich dadurch genauso flexibel wie ESAWN.

Hinsichtlich der gestellten Anforderungen und den gesetzten Randbedingungen müssen bei ESAWN-2 und ESAWN-NR keine Abstriche gemacht werden. Beide Protokolle unterstützen, wie ESAWN, beliebige Aggregationsfunktionen und bieten probabilistische Authentizität. Betrugsversuche werden mit einstellbarer Wahrscheinlichkeit erkannt.

Energie- und Speicherverbrauch von ESAWN-2 und ESAWN-NR liegen, wie bei ESAWN, in $O(1)$ bezogen auf die Netzwerkgröße. Trotzdem verbraucht ESAWN-

2, wie zu erwarten, durch die höhere Anzahl Zeugen mehr Energie als ESAWN. Dadurch kommt es bei einer größeren Anzahl Angreifern und gleichzeig hohen Sicherheitsansprüchen zu teilweise unbefriedigenden Ergebnissen.

ESAWN-NR kommt zwar mit der gleichen Zahl Zeugen wie ESAWN-2 aus, modifiziert jedoch die Kommunikationswege um „Non-Repdudiation“ zu erreichen. Dadurch lassen sich die Daten deutlich schlechter in Pakete zusammenfassen, als dies bei ESAWN-2 der Fall ist. Dadurch gilt hier, noch mehr als bei ESAWN-2, dass bei einer größeren Anzahl Angreifern und gleichzeig hohen Sicherheitsansprüchen die Energiekosten stark ansteigen.

Ein zweites Problem von ESAWN-NR betrifft die Schlüsselaufdeckung im Fall eines entdeckten Betrugsversuch. Durch die Schlüsselaufdeckung erfahren die beteiligten Knoten von Kommunikationsschlüsseln, die sie für den Normalbetrieb des Netzwerks nicht kennen dürfen. Es bleibt offen, wie nach einer solchen Schlüsselaufdeckung der reguläre Netzbetrieb effektiv wieder aufgenommen werden kann, ohne die betroffenen Knoten mit neuen Schlüssel versorgen zu müssen.

Die insgesamt drei ESAWN-Protokolle ermöglichen es dem Netzwerknutzer, den Sicherheits-Energie-Tradeoff noch flexibler zu bestimmen, als dies mit nur einem Protokoll der Fall war. Sowohl die Wahl eines der Protokolle, als auch die anschließende Wahl der Protokollparameter, haben Einfluss auf die erzielte Sicherheit und den Energieverbrauch. Dadurch kann eine Konfiguration bestimmt werden, die optimal an das konkrete Einsatzszenario des Sensornetz angepasst ist.

A. Erweiterte Simulationen

A.1 Aggregationsfunktion und Energieverbrauch

Die im Rahmen dieser Arbeit durchgeführten Simulationen haben bei der Berechnung des Energieverbrauchs nie die Kosten für die Durchführung der Datenaggregation berücksichtigt. Da die ESAWN-Protokolle auf zusätzlicher Berechnung von Aggregaten basieren, ist es nicht offensichtlich, dass hier kein signifikanter Mehrverbrauch entsteht.

In einem Netzwerk N mit $n = 1000$ und $\delta = 2$ werden bei einer konventionellen Aggregation etwa 600 Aggregationen durchgeführt. Bei ESAWN-2 und ESAWN-NR sind dies dagegen, bei $\bar{k} = 2$ bedingt durch die durchgeführten Mehrheitsentscheide, etwa 2880 Aggregationen oder etwa das 5-fache.

Um festzustellen, ob dieser Mehraufwand in Relation zu den Energiekosten für Funk und Kryptographie von Bedeutung ist, wurden zwei Aggregationsfunktionen als Vergleich herangezogen. Die erste Aggregationsfunktion „[1]“summiert die eingehenden Werte lediglich und ist daher von geringer Komplexität. Die zweite Aggregationsfunktion „[2]“wendet den bereits für die Verschlüsselung der Daten verwendeten RC5-Algorithmus als Aggregationsfunktion an und stellt damit eine recht komplexe Berechnung dar.

Der auf den MICA2-Sensorknoten verwendete Prozessor Atmel ATmega128 verbraucht für die erste Aggregationsfunktion 0,00477μAs und für die zweite 2,6μAs. Das Vergleichen zweier 32 bit Werte kostet 0,00477μAs [4; 3; 35].

Abbildung A.1 zeigt den Mehraufwand durch Berücksichtigung der zwei exemplarischen Aggregationsfunktionen „[1]“und „[2]“im Vergleich zu der bisherigen Annahme einer kostenfreien Aggregation „[0]“in einem exemplarisch gewählten Netzwerk. In dieser Darstellung, die den absoluten Energieverbrauch im gesamten Netzwerk darstellt, liegen die Kurven, welche die Kosten unter Verwendung der Aggregationsfunktionen darstellen, kaum erkennbar über der Vergleichskurve „[0]“. Dies trifft auf alle ESAWN-Protokolle und auch auf das Vergleichsprotokoll AGGN zu.

Da diese Darstellung darüberhinaus keine weiteren Möglichkeiten der Analyse zulässt, greifen die beiden Diagramme in Abbildung A.2 nun explizit den Mehrver-

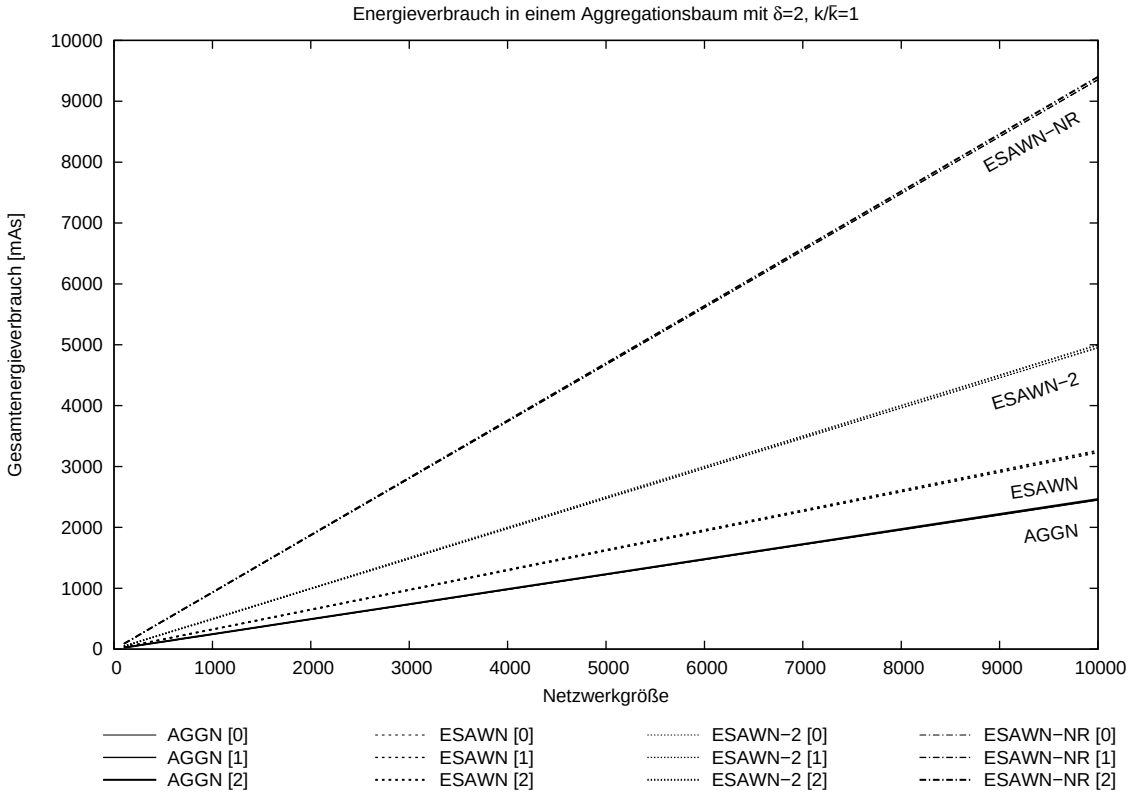


Abbildung A.1: Vergleich des absoluten Energieverbrauchs unter Berücksichtigung verschiedener Aggregationsfunktionen

brauch im Netzwerk auf, der durch die Berücksichtigung einer konkreten Aggregationsfunktion entsteht. Während das obere Diagramm von $k = \bar{k} = 1$ ausgeht, stellt das untere Diagramm die gleiche Situation mit $k = \bar{k} = 2$ dar.

In beiden Diagrammen kann man den Grundaufwand durch das normale Aggregieren bei AGGN ablesen und auch den Zusatzaufwand den die ESAWN-Protokolle durch ihre zusätzlich durchgeführten Aggregationen mit sich bringen. Der Zusatzaufwand ist bei ESAWN-2 und ESAWN-NR nochmals erwartungsgemäß höher als bei ESAWN. Bei $k = \bar{k}$ aggregiert ESAWN jedes Aggregat k mal häufiger als bei AGGN, bei ESAWN-2 und ESAWN-NR dagegen wird jedes Aggregat sogar $2\bar{k}$ mal häufiger als bei AGGN aggregiert.

Im direkten Vergleich beider Diagramme lässt sich zudem gut erkennen, dass der Mehraufwand durch Aggregation mit höherem $k/\bar{k}$ noch einmal deutlich zunimmt. Generell gilt allerdings, dass bei ESAWN-2 und ESAWN-NR der gleiche Mehraufwand anfällt, da sich die Protokolle in diesem Punkt nicht unterscheiden.

Bislang offen bleibt die Frage, ob dieser Mehraufwand einen relevanten Einfluss auf den Gesamtenergieverbrauch der ESAWN-Protokolle besitzt. Setzt man den eben betrachteten Energiemehrverbrauch in Relation zu dem Energieverbrauch mit kostenfreier Aggregationsfunktion ergibt sich folgendes Bild (Abbildung A.3).

Der relative Energiemehrverbrauch ist unabhängig von der Netzwerkgröße. Mit größerem k bzw. $\bar{k}$ wird er außerdem nicht größer sondern tendenziell kleiner.

Bei ESAWN bleibt der relative Mehrverbrauch beim Schritt von $k = 1$ auf $k = 2$ etwa konstant, da sich die zusätzlichen Aggregationen und der erhöhte Kommunikationsaufwand die Waage halten. Bei ESAWN-2 und, noch deutlicher, bei ESAWN-NR nimmt der Anteil beim Schritt von $\bar{k} = 1$ auf $\bar{k} = 2$ sogar deutlich ab, da hier die Energiekosten für die zusätzliche Kommunikation durch das größere $\bar{k}$ die Energiekosten für zusätzlich notwendigen Aggregationen übersteigen.

Für die einfache Aggregationsfunktion „[1]“ kann damit festgehalten werden, dass der Anteil am Energieverbrauch, der durch die Aggregationen verursacht wird, stets unter 0.01% liegt. Auch für die sehr komplexe Aggregationsfunktion „[2]“ übersteigt der Anteil nie 1%. Damit kann die Annahme einer kostenfreien Aggregationsfunktion für die Simulationen im Hauptteil dieser Arbeit als unkritisch angesehen werden.

A.2 Vereinfachung des GloMoSim-Modells

Die verwendete GloMoSim-Implementierungen der ESAWN-Protokolle verzichtet auf die Simulation der unteren Netzwerkschichten, da Latenzzeiten und Paketverlustraten für die Bewertung der Protokolle in der Evaluation ohne Belang sind. Wichtig bei dieser Modifikation ist jedoch, dass die erhobenen Daten, wie beispielsweise die Zahl der versendete Datenpakete, nicht beeinflusst werden. Das heisst, diese Ergebnisse sind weiterhin zuverlässig. In einigen Simulationen wurde deshalb zum Test die Simulation der unteren Schichten wieder aktiviert und die Ergebnisse dieser Testläufe mit den vorherigen Ergebnissen verglichen.

Es zeigt sich, dass eine Korrelation zwischen den physikalisch versendeten Pakete und den auf Anwendungsebene gezählten Datenpaketen existiert, die den Schluss zulässt, dass die Modifikation zulässig ist. Dabei stört auch nicht, dass die Zahl

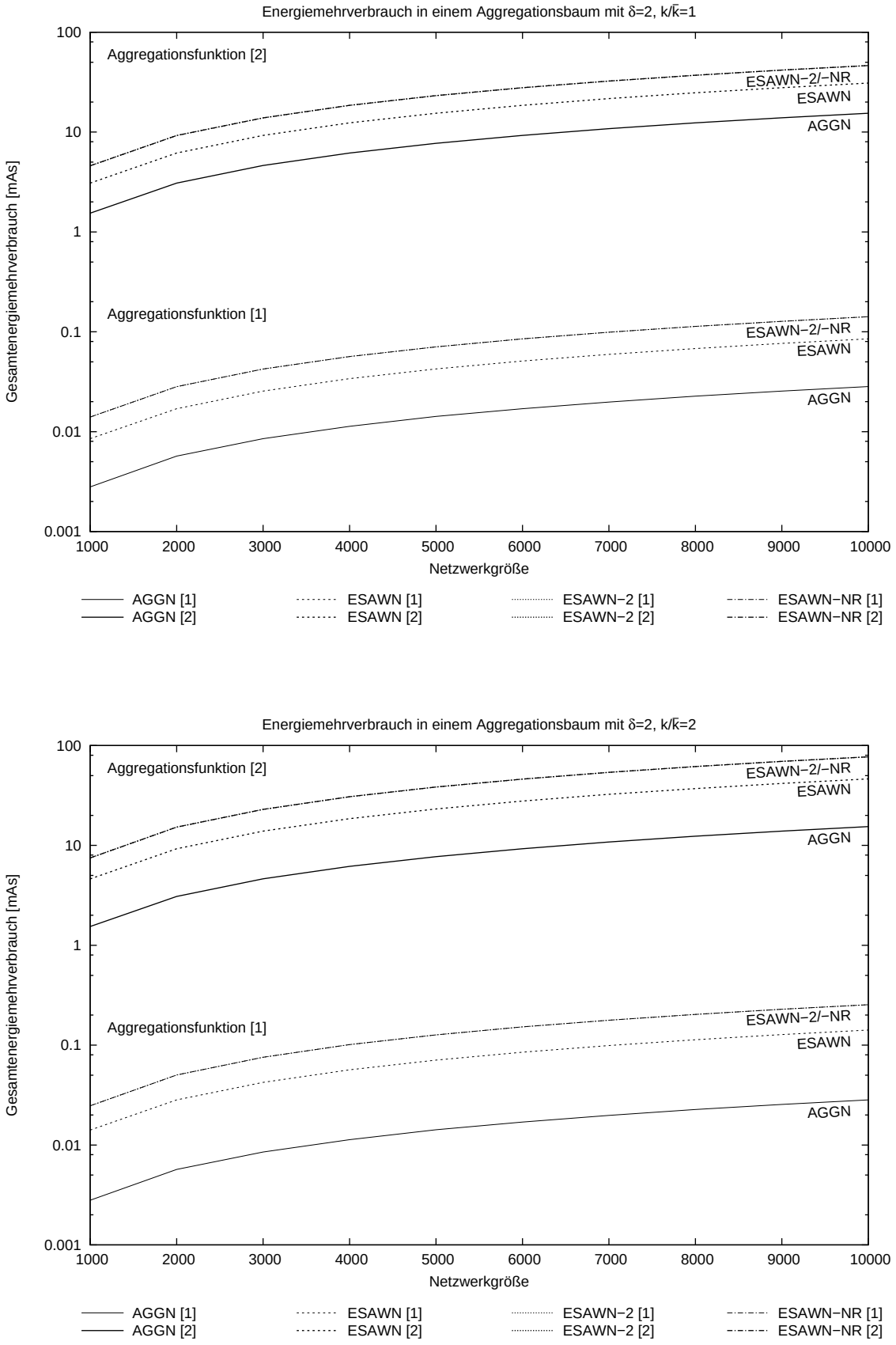


Abbildung A.2: Energiemehrverbrauch durch Berücksichtigung zweier konkreter Aggregationsfunktionen für $k/\bar{k} = 1$ (oben) und für $k/\bar{k} = 2$ (unten)

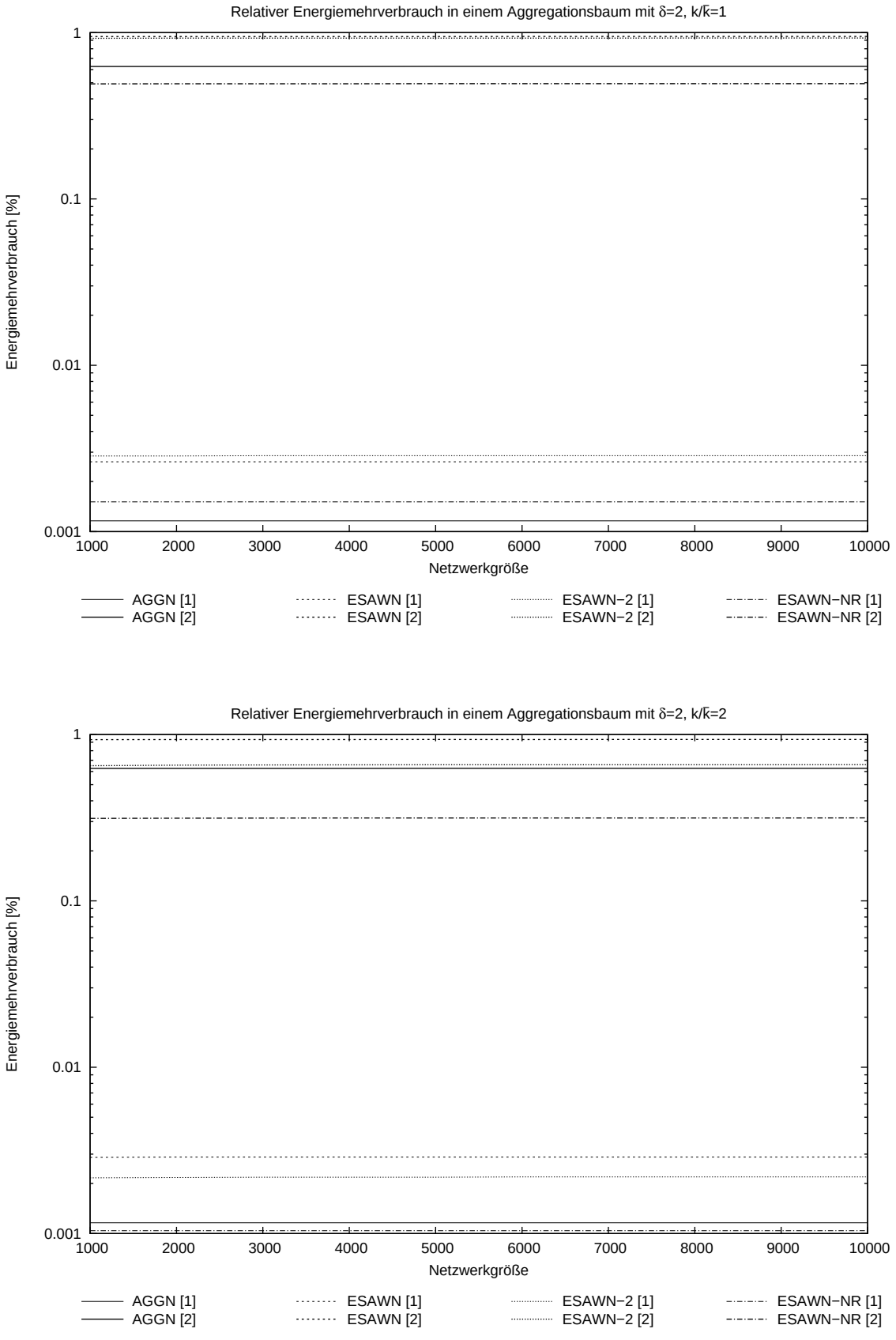


Abbildung A.3: Prozentualer Anteil des Energieverbrauchs durch Aggregation und Datenvergleich am Gesamtenergieverbrauch für $k/\bar{k} = 1$ (oben) und für $k/\bar{k} = 2$ (unten)

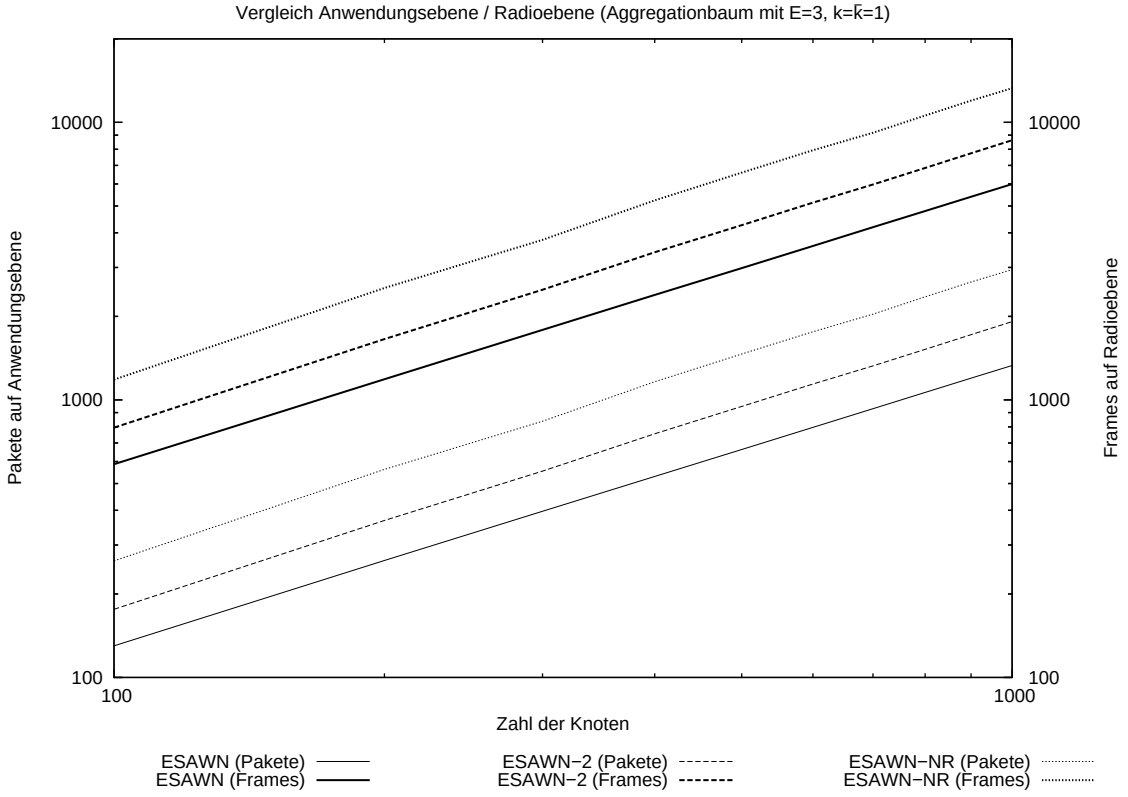


Abbildung A.4: Vergleich der Zahl der versandten Datenpakete auf Anwendungsebene und der tatsächlich physisch versandten Pakete

der physisch versendeten Pakete höher ist, als die auf Anwendungsebene gezählten Pakete. Dies liegt daran, dass Rückbestätigungspakete und verlorene Pakete auf Anwendungsebene nicht berücksichtigt werden. Das Diagramm in Abbildung A.4 veranschaulicht den Zusammenhang der beiden Zählweisen.

Dem Diagramm lässt sich entnehmen, dass es einen direkten Zusammenhang zwischen den Paketen auf Anwendungsebene und den versendeten Datenframes gibt. Der Faktor liegt bei etwa 4,5. Dieser ergibt sich, wenn man bedenkt, dass jedes Datenpaket auf Anwendungseben in vier Datenframes resultiert. Zusätzlich zu dem eigentlichen Datenframe wird zuvor ein RTS- und CTS-Frame generiert. Nach dem Datenpaket wird der Empfang abschließend mit einem ACK-Frame bestätigt. Die verbleibenden 0,5 Datenframes resultieren aus verlorenen oder beschädigten Frames (zum Beispiel durch ein schlechtes Empfangssignal oder Kollisionen). Die hier vorliegenden Ergebnisse basieren auf einer angenommenen Bitfehlerrate von 10^{-6} die in GloMoSim so eingestellt wurde.

Zusammenfassend lässt sich festhalten, dass der Verzicht auf die Simulation der unteren Schichten in den restlichen Simulationen legitim ist und die Evaluation der ESAWN-Protokolle nicht beeinträchtigt.

A.3 Verwendung von SecuSim

Eine Reihe von Simulationsergebnissen in dieser Ausarbeitung wurden, aus Geschwindigkeitsgründen, mit Hilfe von SecuSim (siehe Abschnitt 5.1) gewonnen. Um

	ESAWN		ESAWN-2		ESAWN-NR	
n	$k=1$	$k=2$	$\bar{k}=1$	$\bar{k}=2$	$\bar{k}=1$	$\bar{k}=2$
1000	1,89%	1,52%	1,79%	0,45%	1,20%	2,54%
2000	1,65%	1,74%	1,56%	1,54%	0,75%	1,71%
3000	1,72%	1,63%	1,57%	1,18%	0,76%	1,16%
4000	1,75%	1,82%	1,57%	1,05%	0,89%	1,51%
5000	1,72%	1,65%	1,83%	1,22%	0,77%	1,34%
6000	1,75%	1,61%	1,51%	0,87%	1,33%	1,69%
7000	1,42%	1,60%	1,59%	1,17%	1,14%	1,94%
8000	1,83%	1,34%	1,58%	1,18%	0,93%	1,35%
9000	1,33%	1,52%	1,49%	1,26%	0,65%	0,48%
10000	1,70%	1,72%	1,59%	1,12%	0,77%	0,86%

Tabelle A.1: Relativen Abweichungen der Ergebnisse von GloMoSim und SecuSim

dieses Vorgehen zu legitimieren, wurden die Simulationsergebnisse von SecuSim stichprobenartig mit denen der GloMoSim-Implementierung gegengeprüft.

Das nachfolgende Beispiel mit $n = 1000, \ldots, 10000$, $\delta = 2$ und $k = \bar{k} = 1, 2$ zeigt die Gleichwertigkeit der Ergebnisse von GloMoSim und SecuSim. Betrachtet wurde der Energieverbrauch des Protokolls im Gesamtnetzwerk in As.

In Tabelle A.2 sind die Ergebnisse von GloMoSim wiedergegeben. Tabelle A.3 zeigt die SecuSim-Ergebnisse. Die relativen Abweichungen der Ergebnisse zeigt Tabelle A.1 auf. Insgesamt kann man eine etwa einprozentige Diskrepanz der Ergebnisse beobachten, welche die Auswertung der Ergebnisse nicht wesentlich beeinflusst.

	ESAWN		ESAWN-2		ESAWN-NR	
n	$k=1$	$k=2$	$\bar{k}=1$	$\bar{k}=2$	$\bar{k}=1$	$\bar{k}=2$
1000	328,31	497,61	501,7	1150,42	918,47	2325,1
2000	655,82	998,91	1003,63	2345	1851,7	4746,03
3000	984,8	1497,82	1506,96	3514,63	2780,63	7187,53
4000	1313,95	2001,37	2010,26	4686,49	3704,53	9568,44
5000	1642,03	2497,95	2519,73	5872,89	4637,98	11995,95
6000	1971,18	2996,91	3014,65	7026,58	5535,63	14354,68
7000	2292,4	3496,04	3520,28	8225,41	6471,43	16715,76
8000	2630,45	3985,48	4023,44	9404,13	7412,92	19225,43
9000	2944,89	4492,04	4522,34	10591,11	8364,32	21827,31
10000	3284,32	5001,32	5030,21	11753,52	9282,21	24164,58

Tabelle A.2: Simulationsergebnisse von GloMoSim

	ESAWN		ESAWN-2		ESAWN-NR	
n	$k=1$	$k=2$	$\bar{k}=1$	$\bar{k}=2$	$\bar{k}=1$	$\bar{k}=2$
1000	322,22	490,18	492,86	1145,21	929,61	2385,63
2000	645,18	981,85	988,21	2309,41	1865,64	4828,73
3000	968,15	1473,77	1483,68	3473,66	2801,8	7272,21
4000	1291,36	1965,51	1979,15	4637,8	3737,89	9715,49
5000	1614,32	2457,37	2474,5	5802,05	4673,92	12158,9
6000	1937,29	2949,29	2969,85	6966,31	5610,27	14602,12
7000	2260,25	3441,02	3465,26	8130,5	6546,36	17045,73
8000	2583,22	3932,88	3960,68	9294,69	7482,27	19488,63
9000	2906,18	4424,68	4456,15	10458,95	8418,68	21931,98
10000	3229,33	4916,54	4951,5	11623,14	9354,58	24374,95

Tabelle A.3: Simulationsergebnisse von SecuSim

B. Notation

Die Tabelle B.1 gibt einen Überblick über die in dieser Arbeit verwendete Notation. Neben einer kurzen Beschreibung ist stets einen Verweis auf den Abschnitt der Arbeit aufgeführt, in der das entsprechende Symbol zum ersten Mal Verwendung findet.

Symbol	Beschreibung	Details
$\mathcal{N}$	Menge der Knoten des Netzwerks.	2.3.2.1
$\mathcal{B}$	Menge der korrumpierten Knoten des Netzwerks.	2.3.2.1
n	Zahl der Sensorknoten im Netzwerk. $n \in \mathbb{N}$	2.3.1
δ	Mittlerer Verzweigungsgrad des Aggregationsbaums. Ein Aggregationsknoten besitzt im Mittel δ Kindknoten. $\delta \in \mathbb{N}$	2.2
h	Höhe des Aggregationsbaums. $h \in \mathbb{N}$	2.2
β	Anteil der korrumpierten Knoten im Netzwerk. $\beta \in \mathbb{Q}$, $0 \leq \beta < 1$	2.3.2.1
k	Systemparameter des ESAWN-Protokolls: Zahl der maximal hintereinanderliegenden korrumpierten Knoten auf dem Aggregationspfad. $k \in \mathbb{N}$.	2.6.2
$\bar{k}$	Systemparameter von ESAWN-2 und ESAWN-NR: Zahl der in $2\bar{k}+1$ hintereinanderliegenden Knoten maximal korrumpierten Knoten. $\bar{k} \in \mathbb{N}$.	3.1.1
p	Systemparameter aller ESAWN-Protokolle: Bezeichnet die Wahrscheinlichkeit, dass ein Aggregationsknoten durch zusätzliche Aggregatsberechnung überprüft wird. $p \in \{\frac{i}{100} \mid i \in \mathbb{N} \wedge 0 \leq i \leq 100\}$	2.6.2
$E_{x,y}(\ldots)$	Bezeichnet die Verschlüsselung der in Klammern bezeichneten Daten mit dem symmetrischen Schlüssel, den sich die Knoten x und y teilen.	2.6.2
d_x	Bezeichnet das von Knoten x versendete Datum (Messwert oder Aggregat).	2.6.2
d_x^y	Bezeichnet das von Knoten y empfangene Datum von Knoten x. Dies kann sich ggf. von dem ursprünglich versandten Datum d_x unterscheiden.	3.2.1
agg_x^0	Bezeichnet das Aggregat bei Knoten x, das von Knoten x selbst berechnet wurde.	3.1.2
agg_x^n	Bezeichnet das Aggregat bei Knoten x, das von seinem n-ten Zeugen berechnet wurde.	3.1.2
agg'_x	Bezeichnet ein durch einen Zeugen berechnetes Kontrollaggregat bei ESAWN.	2.6.2

Tabelle B.1: Übersicht über die verwendeten Symbole und Abkürzungen

C. Optimierung des Speicherverbrauchs

Zwei Faktoren haben Einfluss auf den Speicherverbrauch eines Knotens, die Zahl der Zeugen k sowie der Verzweigungsgrad δ des Baumes. Immer dann, wenn ein Knoten mehr als eine Berechnung durchführt, lässt sich Speicherplatz sparen, indem nach jeder Berechnung nicht mehr benötigte Daten verworfen werden. Die folgenden Beispiele sollen verdeutlichen und illustrieren, wie sich eine geschlossene Formel für den optimierten Speicherverbrauch herstellen lässt.

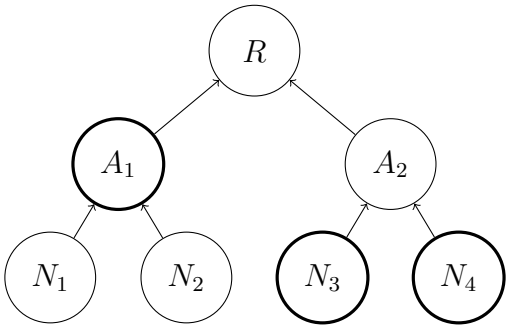
Zuerst soll das Netzwerk aus Abbildung C.1 betrachtet werden, bei dem $\delta = 2$ und $k = 1$ gelte. R hat hier zwei Funktionen, zum einen ist R selbst Aggregationsknoten, zum anderen Zeuge für die Aggregationsknoten A_1 und A_2. R erhält von diesen Knoten zwei Nachrichten mit Messwerten und Aggregaten:

$A_1 \to R: \quad d_1,\, d_2,\, agg_{A_1}$
$A_2 \to R: \quad d_3,\, d_4,\, agg_{A_2}$

Bei der Verarbeitung dieser Nachrichten benötigt R maximal Speicher für drei Messwerte, wie sich aus der Betrachtung des Speicherverbrauchs im unteren Teil der Abbildung C.1 ergibt. Der Zeitpunkt der maximalen Speicherbelegung (Schritt 4) ist dabei eingerahmt und die entsprechenden Knoten sind im Aggregationsbaum ebenfalls hervorgehoben. Neben der Tabelle der Speicherbelegung sind die zugehörigen Schritte beschrieben, die von R zu diesem Zeitpunkt durchgeführt werden. do_agg(...) bezeichnet dabei das Durchführen der Aggregation mit den als Parameter aufgeführten Eingangsdaten.

C.1 Einfluss von δ

Aus diesem Beispiel lassen sich der Einfluss von δ und k auf den Speicherplatzbedarf ableiten: In Schritt 4 wird agg'_{A_1} verworfen, agg_{A_1} aber behalten. Dies liegt daran, dass agg_{A_1} nicht nur für die Verifikation des Knotens A_1 benötigt wird, sondern auch für die Aggregatsberechnung agg_R am Schluss. Erhöht sich jetzt der Verzweigungsgrad auf beispielsweise $\delta = 4$, so müssen zwei weitere Aggregate bis zum Ende der



1.	d_1	d_2			Nachricht 1: d_1, d_2 einlesen.
2.	agg'_{A_1}				$\text{do\_agg}(d_1, d_2) = \text{agg}'_{A_1}$ d_1, d_2 verwerfen.
3.	agg'_{A_1}	agg_{A_1}			Nachricht 1: agg_{A_1} einlesen. Verifikation: $\text{agg}_{A_1} = \text{agg}'_{A_1}$.
4.	agg_{A_1}	d_3	d_4		agg'_{A_1} verwerfen. Nachricht 2: d_3, d_4 einlesen.
5.	agg_{A_1}	agg'_{A_2}			$\text{do\_agg}(d_3, d_4) = \text{agg}'_{A_2}$. d_3, d_4 verwerfen.
6.	agg_{A_1}	agg'_{A_2}	agg_{A_2}		Nachricht 2: agg_{A_2} einlesen. Verifikation: $\text{agg}'_{A_2} = \text{agg}_{A_2}$.
7.	agg_{A_1}	agg_{A_2}			$\text{do\_agg}(\text{agg}_{A_1}, \text{agg}_{A_2}) = \text{agg}_R$. agg_{A_1}, agg_{A_2} verwerfen.
8.	agg_R				

Abbildung C.1: Beispielnetzwerk und Speicherverbrauch bei $\delta = 2$ und $k = 1$.

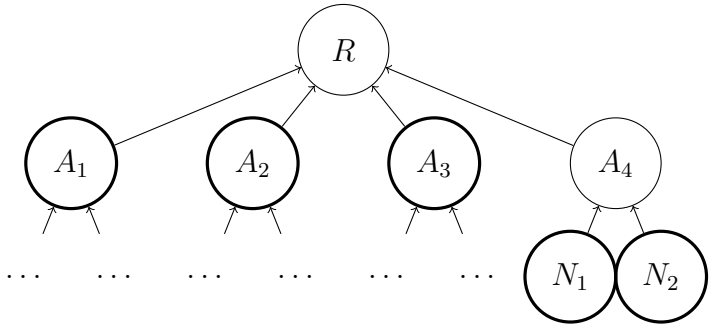


Abbildung C.2: Beispielnetzwerk bei $\delta = 4$ bzw. $\delta = 2$ und $k = 1$.

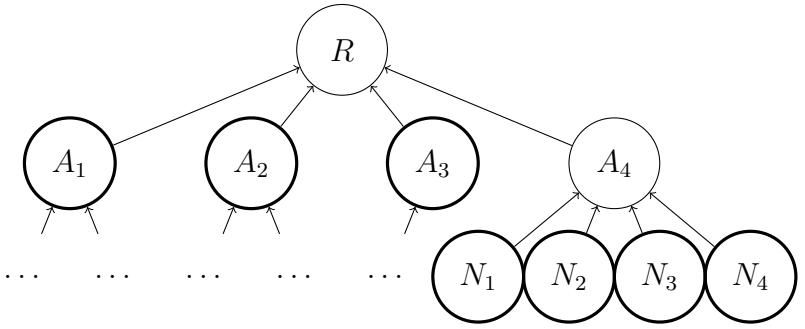


Abbildung C.3: Beispielnetzwerk und Speicherverbrauch bei $\delta = 4$ und $k = 1$.

Berechnung gespeichert bleiben. Der maximale Speicherplatzbedarf steigt also von drei auf fünf Werte, siehe Abbildung C.2. Unterstellt man als Eingangsreihenfolge der Pakete, dass zuerst das Datenpaket von A_1, dann von A_2, usw. eintrifft, kommt es beim Bearbeiten des letzten Datenpakets zur maximalen Belegung des Speichers. Zu diesem Zeitpunkt werden noch agg_{A_1}, agg_{A_2} und agg_{A_3} für die abschließende Aggregation im Speicher gehalten und A_4 wird gerade überprüft.

Bei einem gleichmäßigen Baum steigt der Platzbedarf sogar auf sieben Werte, da nun auch je vier Blattknoten aggregiert werden müssen (Abbildung C.3). Allgemein lässt sich der Platzbedarf bei einem Zeugen mit $(\delta - 1) + \delta$ berechnen.

C.2 Einfluss von k

Auch die Zeugenzahl k hat einen Einfluss auf den maximalen Platzbedarf. Ausgehend von Beispiel 1 bewirkt eine Zeugenzahl von $k = 2$ eine Erhöhung des maximalen Speicherplatzbedarfs auf vier. Der Grund ist die zusätzliche Knotenebene, die jetzt Teil der Berechnungen wird (in Beispiel aus Abbildung C.4 sind dies die Knoten B_x). Allgemein lässt sich der Platzbedarf jetzt mit $k(\delta - 1) + \delta$ berechnen.

C.3 Optimierter Speicherverbrauch

Durch das frühzeitige Verwerfen von nicht mehr benötigten Ergebnissen lassen sich Einsparungen erzielen. Der Speicherverbrauch von ESAWN sinkt von $\frac{\delta^2 \cdot \delta^k - 1}{\delta - 1}$ auf $\delta + k(\delta - 1)$ Werte.

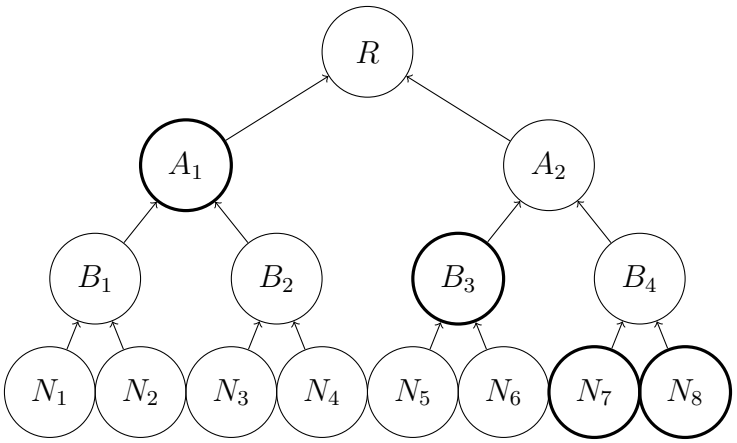


Abbildung C.4: Einfluss der Zeugenzahl auf den Speicherverbrauch.

C.4 Einfluss der Eingangsreihenfolge der Nachrichten

Bisher wurde davon ausgegangen, dass R die Nachrichten immer in Reihenfolge von A_1, A_2, ... erhält. Ist dies nicht der Fall kann dies auch Einfluss auf den maximalen Speicherbedarf haben, vorausgesetzt der Aggregationsbaum ist nicht gleichmäßig aufgebaut, wie im Beispiel aus Abbildung C.5 mit $k = 1$.

R erhält hier zwei Nachrichten:

$A_1 \rightarrow R: \quad d_1, agg_{A_1}$
$A_2 \rightarrow R: \quad d_2, d_3, d_4, agg_{A_2}$

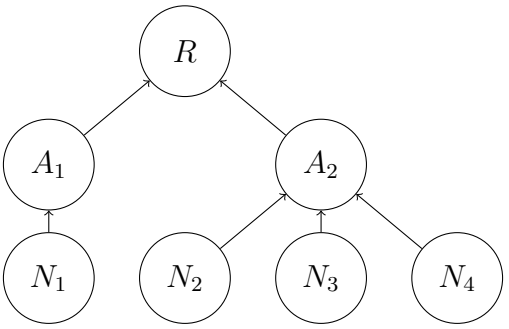
Abhängig von der Bearbeitungsreihenfolge liegt hier der maximale Speicherbedarf bei 3 oder auch 4 Werten. Es kann also Bearbeitungsreihenfolgen geben, die für den Speicherverbrauch besonders günstig sind.

C.5 Optimierung durch Eingangsreihenfolge

Offensichtlich lässt sich der Speicherverbrauch von ESAWN durch eine modifizierte Eingangsreihenfolge der Pakete optimieren. Dabei sind Pakete, die sehr viele Werte enthalten gegenüber kürzeren Transmissionen vorzuziehen. Da diese mehr Daten mit sich führen, brauchen sie zur Aggregation mehr Speicherbedarf als kleinere Pakete. Erfolgt die Bearbeitung dieser Pakete zuerst, bevor andere Daten bereits Teile des Speichers belegen, sinkt die maximale Speicherbelegung, da nach der Aggregation der Daten, ein Großteil des Speichers wieder freigegeben werden kann. Problematisch dabei ist jedoch die Synchronisation der Knoten untereinander um eine möglichst günstige Reihenfolge zu erhalten.

C.6 Transfer auf ESAWN-2 und ESAWN-NR

Das Prinzip des frühzeitigen Verwerfens nicht mehr benötigter Daten, die Auswirkungen des Einflusses der Eingangsreihenfolge der Nachrichten und das mit beiden Ideen verbundene Optimieren des Speicherbedarfs sind nicht auf ESAWN beschränkt. Beides lässt sich auf die neuen Protokolle ESAWN-2 und ESAWN-NR übertragen und auch dort zur Optimierung des Speicherbedarfs nutzen.



1.	d_1			
2.	agg'_{A_1}			
3.	agg'_{A_1}	agg_{A_1}		
4.	agg_{A_1}	d_2	d_3	d_4
5.	agg_{A_1}	agg'_{A_2}		
6.	agg_{A_1}	agg'_{A_2}	agg_{A_2}	
7.	agg_{A_1}	agg_{A_2}		
8.	agg_R			

1.	d_2	d_3	d_4	
2.	agg'_{A_2}			
3.	agg'_{A_2}	agg_{A_2}		
4.	agg_{A_2}	d_1		
5.	agg_{A_2}	agg'_{A_1}		
6.	agg_{A_2}	agg'_{A_1}	agg_{A_1}	
7.	agg_{A_2}	agg_{A_1}		
8.	agg_R			

Abbildung C.5: Relevanz der Bearbeitungsreihenfolge bei ungleichmäßigen Bäumen.

Die Angaben zum Speicherverbrauch in Abschnitt 5.3.4 für ESAWN-2 und in Abschnitt 5.3.5 für ESAWN-NR sind damit als Obergrenzen zu sehen, die durch eine geschickte Implementierung durchaus noch verbessert werden können.

Literatur

1 Acharya, Mithun ; Girao, Joao ; Westhoff, Dirk: Secure Comparison of Encrypted Data in Wireless Sensor Networks. In: *3rd International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks (WiOpt 2005)*. Trentino, Italy : IEEE Computer Society, April 2005. – ISBN 0–7695–2267–X, S. 47–53

2 Anderson, Ross ; Bergadano, Francesco ; Crispo, Bruno ; Lee, Jong-Hyeon ; Manifavas, Charalampos ; Needham, Roger: A new family of authentication protocols. In: *SIGOPS Operating Systems Review* 32 (1998), Nr. 4, S. 9–20. – ISSN 0163–5980

3 Atmel Cooperation: *AVR instruction set – User guide*. http://www.atmel.com/dyn/resources/prod_documents/doc0856.pdf, 10 2005. – Rev. 0856E

4 Atmel Cooperation: *ATmega128(L) reference*. http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf, 10 2006. – Rev. 2467O

5 Becher, Alexander ; Benenson, Zinaida ; Dornseif, Maximillian: Tampering with motes: Real-world physical attacks on wireless sensor networks. In: *Proceedings of International Conference on Security in Pervasive Computing*. York, UK : Springer, Apr 2006. – ISBN 3–54033376–2, S. 104–118

6 Blass, Erik-Oliver: *Sicherer, aggregierender Datentransport in drahtlosen Sensornetzen*. Karlsruhe, Germany : Dissertation, Universitätsverlag Karlsruhe, 2007. – 238 S. – ISBN 978–3–86644–142–2

7 Blass, Erik-Oliver ; Wilke, Joachim ; Zitterbart, Martina: A Security–Energy Trade-Off for Authentic Aggregation in Sensor Networks. In: *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. Washington D.C., USA : IEEE Computer Society, Sep 2006. – ISBN 1–42440–732–X, S. 135–137

8 Blass, Erik-Oliver ; Zitterbart, Martina: An Efficient Key Establishment Scheme for Secure Aggregating Sensor Networks. In: *ACM Symposium on Information, Computer and Communications Security*. Taipei, Taiwan, Mar 2006. – ISBN 1–59593–272–0, 303–310

9 Carman, David W. ; Kruus, Peter S. ; Matt, Brian J.: Constraints and Approaches for Distributed Sensor Network Security. NAI Labs, The Security Research Division, Sep 2000. – Forschungsbericht

10 Castelluccia, Claude ; Mykletun, Einar ; Tsudik, Gene: Efficient Aggregation of Encrypted Data in Wireless Sensor Networks. In: *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. San Diego, CA, USA, Juli 2005. – ISBN 0–7695–2375–7, S. 109 – 117

11 Cramer, Ronald ; Damgard, Ivan: Secure Multiparty Computation, an Introduction. (2003)

12 Crossbow Technology Incorporated: *Crossbow Wireless Sensor Networks – MICAz 2.4GHz*. http://www.xbow.com/Products/productdetails.aspx?sid=164,

13 Dimitriou, Tassos ; Krontiris, Ioannis ; Nikakis, Fotios: Secure In-Network Processing in Sensor Networks. In: Xiao, Prof. Y. (Hrsg.): *Security in Sensor Networks*, CRC Press, 2004, S. 275–290

14 Eschenauer, L. ; Gligor, V.: A key management scheme for distributed sensor networks. In: *Proceedings of ACM Computer and Communications Security*. Washington D.C. USA : ACM Press, Nov 2002. – ISBN 1–58113–612–9, S. 41–47

15 Estrin, Deborah ; Govindan, Ramesh ; Heidemann, John S. ; Kumar, Satish: Next Century Challenges: Scalable Coordination in Sensor Networks. In: *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, S. 263–270

16 Gura, Nils ; Patel, Arun ; Wander, Arvinderpal ; Eberle, Hans ; Shantz, Sheueling C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In: *Lecture Notes in Computer Science* Sun Microsystems Laboratories, Springer Verlag, 2004. – ISSN 0302–9743, S. 119–132

17 Haeberlen, Andreas ; Kouznetsov, Petr ; Druschel, Peter: The Case for Byzantine Fault Detection. In: *Proceedings of the Second Workshop on Hot Topics in System Dependability (HotDep'06)*. Berkeley, CA, USA : USENIX Association, Nov 2006

18 Hu, Lingxuan ; Evans, David: Secure Aggregation for Wireless Networks. In: *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*. Washington, DC, USA : IEEE Computer Society, Jan 2003. – ISBN 0–7695–1873–7, S. 384

19 Intanagonwiwat, Chalermek ; Estrin, Deborah ; Govindan, Ramesh ; Heidemann, John: Impact of network density on data aggregation in wireless sensor networks. In: *Proceedings of International Conference and Distributed Computing Systems*, IEEE Computer Society, Jul 2002. – ISBN 0–7695–1588–6, S. 457–458

20 Intanagonwiwat, Chalermek ; Govindan, Ramesh ; Estrin, Deborah ; Heidemann, John ; Silva, Fabio: Directed diffusion for Wireless Sensor Networking. In: *ACM Transactions on Networking*, 2003. – ISBN 1–06366–9, S. 2–16

21 International Organization for Standardization: *ISO/IEC 13888-2. Information technology – Security techniques – Non-repudiation: mechanisms using symmetric techniques*. 1998

22 International Organization for Standardization: *ISO/IEC 13888-1. Information technology – Security techniques – Non-repudiation: general*. 2004

23 Kahn, Joseph M. ; Katz, Randy H. ; Pister, Kristofer S. J.: Emerging Challenges: Mobile Networking for „Smart Dust“. In: *J. Comm. Networks* (2000), September, S. 188–196

24 Lamport, Leslie ; Shostak, Robert ; Pease, Marshall: The Byzantine Generals Problem. In: *Advances in Ultra-Dependable Distributed Systems*. N. Suri, C. J. Walter, and M. M. Hugue. IEEE Computer Society Press, Jan 1995. – ISBN 0–8186–6287–5, S. 382 – 401

25 Perrig, Adrian ; Stakovic, John ; Wagner, David: Security in Wireless Sensor Networks. In: *Communications of the ACM* 47 (2004), Juni, Nr. 6, S. 53–57. – ISSN 0001–0782

26 Perrig, Adrian ; Szewczyk, Robert ; Wen, Victor ; Culler, David E. ; Tygar, J. D.: SPINS: security protocols for sensor networks. In: *Mobile Computing and Networking*, 2001, S. 189–199

27 Przydatek, Bartosz ; Song, Dawn ; Perrig, Adrian: SIA: secure information aggregation in sensor networks. In: *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA : ACM Press, 2003. – ISBN 1–58113–707–9, S. 255–265

28 Qadi, Ala: *Object Tracking Using Sensor Networks (Report)*. http://cse.unl.edu/~sdas/WSNL/report.pdf,

29 Raghunathan, Vijay ; Schurgers, Curt ; Park, Sung ; Srivastava, Mani B.: Energy-Aware Wireless Microsensor Networks. In: *IEEE Signal Processing Magazine* 19 (2002), März, Nr. 2, S. 40–50. – ISSN 1053–587X

30 Rogaway, Phillip ; Bellare, Mihir ; Black, John ; Krovetz, Ted: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: *ACM Conference on Computer and Communications Security*, 2000. – ISBN 1–58113–203–4, S. 196–205

31 Sausa, Rosario C.: Real-time, laser-based sensors for military and civilian applications. In: *Proceedings for the Army Science Conference*, 2004

32 Schmid, Thomas ; Dubois-Ferriere, Henri ; Vetterli, Martin: SensorScope: Experiences with a Wireless Building Monitoring Sensor Network. In: *Workshop on Real-World Wireless Sensor Networks (REALWSN'05)* Ecole Polytechnique Federale de Lausanne (EPFL), Schweiz, 2005

33 Schulze, Christian: *IT-System-Elektroniker Handbuch – Hilfe bei der IT-Ausbildung: Das ISO-OSI-Referenzmodell*. http://www.itse-guide.de/artikel/iso-osi-referenzmodell, 2006

34 UCLA Parallel Computing Laboratory: *Parsec: Parallel Simulation Environment for Complex Systems*. http://pcl.cs.ucla.edu/projects/parsec/, 7 2007

35 University of California: *TinySec: Link Layer Encryption for Tiny Devices*. Berkeley, CA, USA : http://www.cs.berkeley.edu/~nks/tinysec/, August 2006

36 University of California: *TinyOS, an open-source operating system designed for wireless embedded sensor networks*. Berkeley, CA, USA : http://www.tinyos.net/, September 2007

37 Warneke, Brett ; Last, Matt ; Liebowitz, Brian ; Pister, Kristofer S. J.: Smart Dust: Communicating with a Cubic-Millimeter Computer. In: *Computer* 34 (2001), Nr. 1, S. 44–51. – ISSN 0018–9162

38 Welsh, Matt: *CodeBlue: Wireless Sensor Networks for Medical Care*. Cambridge, MA, USA : http://www.eecs.harvard.edu/~mdw/proj/codeblue/, September 2006

39 Westhoff, Dirk ; Girao, Joao ; Schneider, Markus: Concealed Data Aggregation for Reverse Multicast Traffic in Wireless Sensor Networks. In: *Proceedings of the 2006 International Conference on IEEE Transactions on Mobile Computing* Bd. 5, 2006. – ISBN 0–7803–8938–7, S. 3044–3049

40 Wilke, Joachim: *Energiequellen für Sensoren in drahtlosen Sensornetzen*. Jul 2005. – Seminararbeit im Rahmen des Seminars Sensornetze am Institut für Telematik, Universtität Karlsruhe.

41 Wilke, Joachim: *Authentischer und effizienter Datentransport in drahtlosen Sensornetzen*. 2006. – Studienarbeit am Institut für Telematik, Universtität Karlsruhe.

42 Zeng, Xiang ; Bagrodia, Rajive ; Gerla, Mario: GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In: *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*. Banff, Alberta, Canada, Mai 1998. – ISSN 0163–6103, S. 154–161

43 Zhu, Sencun ; Setia, Sanjeev ; Jajodia, Sushil: LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In: *ACM Conference on Computer and Communications Security (CCS '03)*, 2003. – ISBN 1–58113–738–9, S. 62–72

44 Zhu, Sencun ; Setia, Sanjeev ; Jajodia, Sushil ; Ning, Peng: An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks. In: *Proceedings of IEEE Symposium on Security and Privacy, Oakland, California*, 2004. – ISBN 0–7695–2136–3, S. 259–271

Index