Semesterarbeit

Routing in Ad-Hoc-Netzen

Björn Glaus
email: glauss@student.ethz.ch

Departement Informatik
Eidgenössische Technische Hochschule Zürich
21. August 2002

Prof. Dr. Roger Wattenhofer
Distributed Computing Group
Betreuer: Aaron Zollinger

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit behandelt Routing in mobilen Ad-Hoc-Netzen. Unter Ad-Hoc-Netzen versteht man sich selbst organisierende Kommunikationsnetze, in denen mobile Geräte ohne zentrale Infrastruktur ein temporäres Netz aufbauen und unterhalten können. Ohne zentrale Infrastruktur bedeutet, dass die Verwaltung des Netzes so verteilt ist, dass Administrationsaufgaben von den Kommunikationsteilnehmern selber übernommen werden. Eine solche Aufgabe ist beispielsweise das richtige Routen von Nachrichtenpaketen über mehrere Stationen hinweg (Multi-Hop-Routing). In Ad-Hoc-Netzen sind die mobilen Stationen also auch die Router. Erschwerend kommt hinzu, dass die Topologie des Netzes sehr dynamisch sein kann: Einerseits, weil die Stationen mobil sind, andererseits durch die geringere Zuverlässigkeit von drahtlosen Verbindungen gegenüber solchen in einem Festnetz. Da für das Routing in Ad-Hoc-Netzen also andere oder anders gewichtete Kriterien gelten, wurden (und werden immer noch) Protokolle eigens für diese Aufgabe entwickelt.

Im ersten Teil der Arbeit wird ein bestehendes Routing-Protokoll für Ad-Hoc-Netze vorgestellt. Danach bespreche und analysiere ich eine mögliche Verbesserung dieses Protokolls.

Der zweite Teil der Arbeit behandelt die physischen Parameter eines Ad-Hoc-Netzes. Die Resultate dieses Teils beruhen auf Messungen, die ich mit mehreren Laptops gemacht habe, die jeweils mit einer ad-hoc-fähigen Wireless LAN-Karte ausgerüstet waren.

2 Dynamic Source Routing

Dynamic Source Routing [4, 5, 6] ist ein reaktives Routing-Protokoll für mobile Ad-Hoc-Netze. Reaktive Protokolle generieren Routen erst wenn notwendig. Die Kernidee des DSR-Protokolls ist der Gebrauch von Source-Routing, d.h. der Sender einer Nachricht kennt die gesamte Hop-By-Hop-Route zum Empfänger. Diese Route wird explizit als Teil des Paketheaders versendet. Dadurch müssen Knoten, die eine Nachricht weiterleiten, keine internen Tabellen konsultieren, um den ausgehenden Pfad zu bestimmen, da dieser ja Teil der Nachricht selbst ist.

Der Hauptvorteil von Source Routing gegenüber konventionellen Routing-Mechanismen, wie z.B. Distance Vector Routing, besteht darin, dass periodischer (proaktiver) Nachrichtenverkehr zur Aufrechterhaltung von Routingtabellen vermieden werden kann. Dadurch wird Energie gespart, wenn sich die Topologie des Netzes wenig verändert oder wenn vorübergehend kaum Nachrichten ausgetauscht werden. Energieeffizienz ist ein wichtiges Kriterium in mobilen Netzen.

Das Protokoll besteht aus zwei Phasen: *Route Discovery* und *Route Maintenance*.

2.1 Route Discovery

Der Sender s möchte ein Nachrichtenpaket dem Empfänger t senden, kennt aber den Pfad zu diesem nicht. Er sendet per Broadcast ein *ROUTE REQUEST*-Paket *RREQ*. Jeder Zwischenknoten n_x, der eine *RREQ*-Nachricht empfängt, konsultiert seinen Cache (siehe 2.1.1). Ist die Route zu t unbekannt, hängt er seine Adresse an die *RREQ*-Nachricht und flutet weiter.

Sobald der Empfänger t erreicht ist oder ein Knoten n_x einen Pfad $(n_x, \ldots, t)$ im Cache hat, wird ein *ROUTE REPLY*-Paket *RREP* generiert, das den gesamten Pfad von s nach t enthält. Falls symmetrische Links unterstützt werden, kann die *RREP* den umgekehrten Pfad der *RREQ* benutzen, sonst muss eine andere Technik gewählt werden [4].

Jeder Knoten auf dem Rückweg von t nach s speichert die Route nach t in seinen Cache.

2.1.1 Caching

Um zu vermeiden, dass bei jedem Paket das Netz erneut geflutet werden muss, werden gelernte Routen in einen Cache gespeichert. Dieser wird in zweierlei Hinsicht gebraucht: Wenn der Senderknoten s bereits ein Paket an den Empfänger t gesendet hat, kann er bei späteren Paketen die Route seinem Cache entnehmen. Kennt der Knoten s den Pfad zu t nicht, flutet er das Netz. Sobald ein Zwischenknoten n_x eine Route nach t in seinem Cache

hat, muss n_x nicht mehr weiterfluten, sondern kann eine $RREP$ mit den zusammengehängten Routen $(s, \ldots, n_x)$ und $(n_x, \ldots, t)$ an s senden.

2.2 Route Maintenance

Die Pfade werden aufrechterhalten mittels $ACKNOWLEDGEMENT$-Paketen ACK, und $ROUTE\ ERROR$-Paketen $RERR$. Eine $RERR$-Nachricht wird erzeugt, wenn ein Knoten feststellt, dass eine Verbindung nicht mehr funktioniert. Diese $RERR$ wird dem Sender s zugesendet. Dieser kann darauf erneut eine *Route Discovery* starten. Die $RERR$-Nachricht veranlasst alle Zwischenknoten inklusive s, die fehlerhafte Route aus dem Cache zu löschen.

3 Verbesserungen

In diesem Kapitel werde ich eine mögliche Verbesserung des DSR-Protokolls besprechen und analysieren. Ein Kommunikationsnetz wird dazu durch einen Graphen modelliert, wobei die Knoten die mobilen Stationen repräsentieren, die Kanten die Erreichbarkeitsrelation zwischen den Stationen. Es wird davon ausgegangen, dass die Verbindungen zwischen den Stationen symmetrisch sind. Die Graphen sind also ungerichtet.

Wie oben beschrieben, besitzt jeder Knoten einen Cache, in dem kürzlich gebrauchte Routen gespeichert werden. Durch das Abspeichern dieser Routen soll das Fluten des Netzes vermindert werden. Grundsätzlich kann ein Sender durch eine *Route Discovery* mehrere – alternative – Pfade zum Empfänger lernen. Durch den Gebrauch dieser alternativen Pfade kann das erneute Fluten im Falle eines Verbindungsfehlers unter Umständen weiter vermindert werden. An diese Beobachtung knüpfen die folgenden Überlegungen an.

3.1 Grundlegende Möglichkeiten

Der Nutzen alternativer Pfade soll durch folgendes Beispiel motiviert werden: In Abbildung 1(a) startet der Sender s eine *Route Discovery*. Unter noch zu erläuternden Umständen erhält s zwei $RREP$ mit den Pfaden $P_1 = (s, n_1, t)$ sowie $P_2 = (s, n_2, t)$. Bewegt sich n_1 zu einem späteren Zeitpunkt weg, stellt s einen Verbindungsfehler fest. Hat s nun nur P_1 in seinem Cache abgelegt, muss das Netz erneut geflutet werden. Als Resultat erhält s den Pfad P_2, der bereits nach dem ersten Fluten bekannt gewesen wäre. Die grundsätzliche Idee besteht also darin, mehrmaliges Fluten des Netzes zu vermeiden, indem mehr Information aus einer *Route Discovery* gewonnen wird.

Bei genauerer Betrachtung zeigt sich, dass zwischen drei Strategien zur Erlangung von alternativen Pfaden unterschieden werden kann. Die drei Möglichkeiten werden im Folgenden kurz vorgestellt und danach genauer analysiert.

3.1.1 Strategie A: Keine Protokolländerung

In der ursprünglichen Beschreibung des DSR-Protokolls [4] behandelt jeder Knoten, inklusive Empfänger, eine $RREQ$ nur einmal. Dies wird anhand des $(Sender, ID)$ -Paares überprüft. Auf diese Weise kann zumindest bei der ersten *Route Discovery* innerhalb des Netzes nur ein Pfad gelernt werden. Wird zu einem späteren Zeitpunkt eine *Route Discovery* gestartet, kann s zu alternativen Pfaden gelangen, falls ein Zwischenknoten n_x bereits einen Pfad $(n_x, \ldots, t)$ im Cache abgelegt hat. Angenommen in Abbildung 1(a) hätte n_2 bereits einen Pfad zu t im Cache: Startet s nun eine *Route Discovery*,

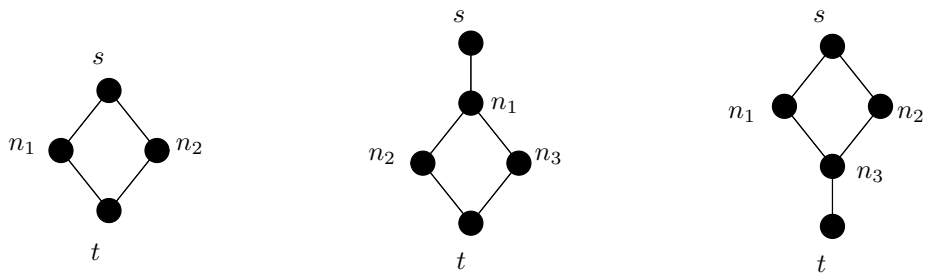


Abbildung 1: Alternative Pfade von s nach t

so beantwortet t die *RREQ*-Nachricht, die über n_1 zu t gelangt ist und n_2 direkt die *RREQ* von s. Auf diese Weise lernt s die beiden Pfade (s, n_1, t) und (s, n_2, t).

3.1.2 Strategie B: Der Empfänger beantwortet alle *RREQ*

Wie oben beschrieben, wird mit dem ursprünglichen Protokoll *höchstens ein* Pfad gelernt, wenn kein Zwischenknoten eine Route im Cache hat. Dies, obwohl durch das Fluten mehrere *RREQ*-Pakete den Empfänger erreichen können, also verschiedene Routen entdeckt werden. Folgende Protokolländerung liegt nun nahe: Der Empfängerknoten beantwortet *alle RREQ*-Pakete, die er empfängt.

Gegeben sei ein Szenario wie in Abbildung 1(b). Durch die Protokolländerung werden (wie auch in Abbildung 1(a)) beide Routen von s nach t erlernt. Es stellt sich nun die Frage, ob n_1 beide Routen an s weiterleiten soll oder nur eine. Falls alle Pfade an s propagiert werden, werden im Fall von späteren *RERR*-Nachrichten auch diese an s weitergeleitet, worauf s dem Cache einen alternativen Pfad entnehmen kann. Der Knoten n_1 kann aber auch nur einen Pfad an s weiterleiten, da Nachrichtenpakete in jedem Fall über n_1 gesendet werden. Erhält n_1 eine *RERR*-Nachricht, so kann n_1 selber versuchen, das Nachrichtenpaket über einen alternativen Pfad zu senden. In diesem Fall muss n_1 aber s mitteilen, dass der alte Pfad nicht mehr gültig ist sowie den neuen zusenden. Diese beiden Optionen werden ebenfalls analysiert.

3.1.3 Strategie C: Mehrmaliges Weitersenden einer *RREQ*

Gegeben sei ein Szenario wie in Abbildung 1(c). Auch in diesem Fall gibt es mehrere Pfade von s nach t. Damit diese Pfade gelernt werden können, muss der Knoten n_3 eine *RREQ*-Nachricht mit einem $(Sender, ID)$-Paar, das er bereits einmal erhalten hat, erneut weitersenden, falls seine eigene Adresse

nicht darin enthalten ist. Der Empfänger beantwortet beide *RREQ*. In diesem Fall muss n_3 natürlich beide *RREP* an s weiterleiten, damit dieser von den alternativen Pfaden Gebrauch machen kann. Es gilt zu beachten, dass das mehrmalige Weitersenden einer *RREQ* in einem Graphen wie in Abbildung 1(b) nicht notwendig ist.

3.2 Analyse

In diesem Abschnitt werden die verschiedenen Varianten analysiert. Dabei interessieren vor allem folgende Punkte:

- Anzahl versendeter *RREQ*-Nachrichten: Es wird davon ausgegangen, dass der Sender mit einer *RREQ*-Nachricht alle seine Nachbarn erreichen kann.
- Anzahl versendeter *RREP*-Nachrichten: Die *RREP*-Nachrichten werden auf der Route zurückgesendet, die im Paketheader der *RREQ* angegeben ist. Die Pakete sind also adressiert und werden einzeln gezählt.
- Benötigte Cachegrössen.
- Grösstmögliche Fehler-Latenz: Wie oft werden im schlimmsten Fall Nachrichtenpakete über alternative Pfade versendet, obwohl keine Verbindung mehr zu t existiert.

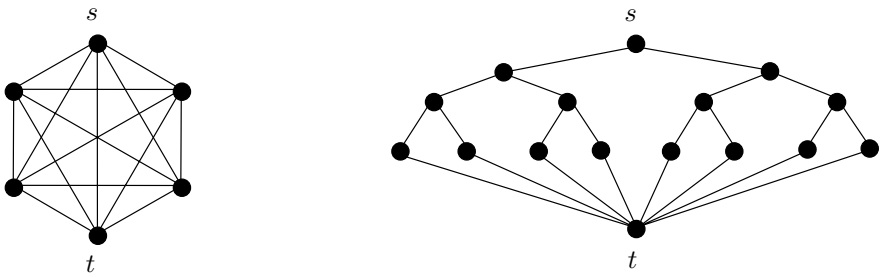
Für diese Kriterien sollen mittels ausgewählter Graphen untere Schranken für den schlechtesten Fall gefunden werden.

3.2.1 Strategie B

Gegeben der Fall, dass ein Zwischenknoten eine *RREQ*-Nachricht nur einmal weitersendet, d.h. über jeden Knoten wird maximal eine *RREQ* gesendet. Man sieht, dass mit dieser Strategie höchstens soviele Pfade gelernt werden können wie t eingehende Kanten hat.

Für die Analyse betrachte ich die beiden Graphen G_1 und G_2, wie sie schematisch in Abbildung 2 dargestellt sind. G_1 ist der vollständige Graph K_n, G_2 besteht aus einem balancierten Binärbaum T sowie dem Empfangsknoten t, der mit allen Blättern von T verbunden ist. Die Knotenanzahl $|V(G_2)| = n$ beträgt 2^l für ein $l \in \mathbf{N}$. Die Kantenanzahl $|E(G_2)| = m$ ist $3 \cdot 2^{l-1} - 2$.

Anzahl *RREQ*-Nachrichten: In beiden Graphen wird von jedem Knoten ausser t genau eine *RREQ* versendet. Es werden also $\Theta(n)$ *RREQ* versendet.



(a) (b)

Abbildung 2: G_1 und G_2

Anzahl *RREP*-Nachrichten: Im Graphen K_n macht es keinen Unterschied, ob alle *RREP* an s propagiert werden oder nicht. In beiden Fällen empfängt s genau $n-1 = \Omega(n)$ *RREP*, da es soviele disjunkte Pfade gibt (jede nach t eingehende Kante ergibt eine Route).

Werden im Graphen G_2 alle *RREP* an s propagiert, so werden im gesamten Netz auf jeder Stufe $deg(t)$ *RREP* versendet, also insgesamt $l \cdot 2^{l-1} = \Omega(n \cdot log(n))$ *RREP*-Nachrichten.

Benötigte Cachegrössen: Wie der Graph K_n zeigt, empfängt s im schlechtesten Fall $\Omega(n)$ *RREP*-Pakete. Natürlich kann in der Praxis diese Zahl auf einen Maximalwert gesetzt werden.

Maximale Latenz: Für dieses Kriterium betrachte ich den Graphen G_2. Für das Versenden eines Datenpaketes über eine Kante rechne ich mit einer Zeitdauer von τ_1. Die Verzögerung, die entsteht bis festgestellt wird, dass die Verbindung $n_x - t$ nicht mehr funktioniert, dauert jeweils τ_2.

Falls die alternativen Pfade von den Zwischenknoten verwaltet werden, wird über jede Kante von T einmal ein Nachrichtenpaket und einmal ein *RERR*-Paket versendet. Zusätzlich entsteht $deg(t)$-mal ein Verbindungsfehler. Das ergibt für G_2 eine maximale Fehler-Latenz von: $\tau_1 \cdot 2(2^l - 2) + \tau_2 \cdot 2^{l-1} = \tau_1 \cdot \Omega(n) + \tau_2 \cdot \Omega(n)$. Es ist wichtig zu bemerken, dass s nach $\frac{deg(t)}{2} = \Omega(n)$ erfolglosen Sendeversuchen überhaupt erst die Gelegenheit hat, einen solchen frühzeitig abzubrechen.

Werden alle *RERR* an s propagiert, so dauert das Versenden aller Datenpakete in T $\tau_1 \cdot 2(l-1) \cdot 2^{l-1}$, da es 2^{l-1} Pfade der Länge $l-1$ zu den Blättern von T gibt. Damit entsteht eine Fehlerlatenz von $\tau_1 \cdot \Omega(n \cdot log(n)) + \tau_2 \cdot \Omega(n)$.

Der Sender hat bei der zweiten Strategie die Möglichkeit, jederzeit eine

Sendewiederholung über alternative Pfade abzubrechen und eine neue *Route Discovery* zu starten.

3.2.2 Strategie C

Ich betrachte den Fall, dass ein Zwischenknoten n_x *RREQ*-Nachrichten, die verschiedene Pfade von s nach n_x genommen haben, weiterleitet. Dies entspricht schematisch der Situation in Abbildung 1(c). Für die Analyse benutze ich einen Graphen $G_3 = (V, E)$, wie er in Abbildung 3 dargestellt ist.[1] Die Knotenanzahl $|V| = n$ beträgt l^2 für ein $l > 1$, die Kantenanzahl $|E| = m$ ist $2l^2 - 2l$.
Es gilt folgende Flooding-Regel:

- Hat n_x bereits eine *RREQ*-Nachricht mit demselben $(Sender, ID)$-Paar erhalten, so sendet er sie nur weiter, wenn der Pfad $(s, \ldots, n_x)$ der neuen *RREQ*-Nachricht nicht grösser als derjenige des alten ist.

Der Grund für diese Regel ist, dass in Abbildung 3(a) beispielsweise der Pfad $(s, n_{1,0}, n_{2,1}, n_{1,1}, \ldots, t)$ nicht gelernt werden soll, d.h. in G_3 werden nur die kürzesten Pfade gelernt. Für Strategien, bei denen auch längere Routen gelernt werden sollen, stellen die folgenden Berechnungen eine untere Schranke dar.

Durch die Flooding-Regel wird auch die Zyklenfreiheit garantiert, da ein Paket mit einem Zyklus immer einen längeren Pfad genommen hat als ein Paket ohne.

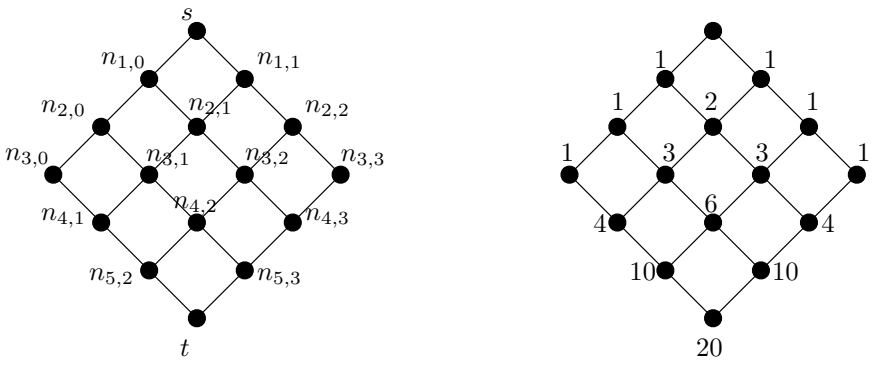
Anzahl *RREQ*-Nachrichten: Mit dieser Strategie werden so viele *RREQ*-Nachrichten versendet, wie kürzeste Pfade $(s, \ldots, t)$ existieren. Die Anzahl kürzester Wege von s zu einem Knoten $n_{i,j}$ ist der Binomialkoeffizient $\binom{i}{j}$ (Abbildung 3(b)). Der Empfänger t erhält also

$$\binom{2(l-1)}{(l-1)} \geq \frac{2^{2(l-1)}}{l} = \Omega(2^{2l} \cdot l^{-1}) = \Omega(2^{2\sqrt{n}} \cdot n^{-1/2})$$

RREQ-Nachrichten. Die Anzahl *RREQ*, die im gesamten Netz versendet werden, beträgt

$$\sum_{i=1}^{l} \binom{l-1+i}{i} - 1 = 2 \cdot \binom{2l-1}{l} - 2 = \binom{2l}{l} - 2 \geq \frac{2^{2l}}{l+1} - 2 = \Omega(2^{2\sqrt{n}} \cdot n^{-1/2})$$

[1] Eine Diskussion mit Aaron Zollinger hat gezeigt, dass der Graph G_3 nicht der schlechtest mögliche ist. Man kann $\frac{\sqrt{n}}{2}$ komplett bipartite Graphen $K_{\sqrt{n},\sqrt{n}}$ verknüpfen und s mit den ersten $\sqrt{n}$ Knoten, t mit den letzten verbinden. Dies ergibt einen Graphen, der $\Omega(\sqrt{n}^{\sqrt{n}})$ kürzeste Pfade besitzt. Dieser Graph besitzt allgemein einfachere Eigenschaften als der von mir gewählte.



(a) (b)

Abbildung 3: Graph G_3

Anzahl *RREP*-Nachrichten: Werden alle *RREP* an s propagiert, wird das Netz quasi rückwärts geflutet, d.h. s erhält $\Omega(2^{2\sqrt{n}} \cdot n^{-1/2})$ *RREP*-Nachrichten. Im Unterschied zu den *RREQ*-Nachrichten wird aber jede *RREP* nur über eine Kante gesendet.

Benötigte Cachegrössen: Da die Anzahl Pfade fast exponentiell mit der Anzahl Knoten steigen kann, gilt dies auch für die Cachegrössen, wenn alle Routen gespeichert werden sollen.

Betrachtet man kurz die Strategie B in Bezug auf G_3, so sieht man, dass nur $deg(t) = 2$ alternative Pfade gefunden werden.

3.2.3 Strategie A

Mit der bisherigen Strategie, dass jeder Knoten inklusive t eine *RREQ*-Nachricht nur einmal behandelt, ist die Anzahl gelernter Routen bei der erstmaligen *Route Discovery* im Netz immer höchstens 1. Ich interessiere mich nun aber für die grösstmögliche Anzahl alternativer Pfade, die ohne eine Protokolländerung gelernt werden kann. Im Graphen G_2 ist dies 2^{l-1}, nämlich dann, wenn jedes Blatt von T bereits einen Pfad zu t kennt. Im Graphen G_3 kann s $2 \cdot (l-1)$ *RREP* erhalten, nämlich dann, wenn alle Knoten an den unteren Quadratseiten eine Route im Cache haben. Die *maximale* Anzahl alternativer Pfade, die ohne Protokolländerung gelernt werden kann, ist nie kleiner als mit der Strategie B (Gleichheit gilt, wenn genau alle Nachbarn von t eine Route zu t kennen).

3.3 Fazit

Die Strategie C, *RREQ*-Nachrichten mehrmals weiterzusenden, ist sehr teuer, wächst doch der Nachrichtenverkehr im schlimmsten Fall fast exponentiell mit der Anzahl Knoten im Netz. Die Strategie B, dass der Empfänger alle *RREQ*-Nachrichten beantwortet, ist im Vergleich um ein Vielfaches billiger: Die Anzahl *RREP*-Nachrichten wächst im gesamten Netz mit $\Omega(n \cdot log(n))$.

Der Vorteil der Strategie B gegenüber der ursprünglichen, wo auch der Empfängerknoten eine *RREQ*-Nachricht nur einmal behandelt, liegt darin, dass mehr Information aus einer *Route Discovery* gewonnen werden kann wenn das Netz neu formiert wird. Dieser Vorteil wird allerdings kleiner, je länger das Netz existiert. Die Nachbarn von t lernen sehr rasch die Route zu t und speichern sie in ihrem Cache. Dadurch werden diese Pfade bei nachfolgenden *Route Discoveries* auch ohne Protokolländerung gelernt.

Bei der Entscheidung, ob alle alternativen Pfade an s zurückgeleitet werden sollen oder jeweils nur einer, sprechen drei Gründe für ersteres:

1. Der Sender hat die Möglichkeit, jederzeit eine Sendewiederholung über alternative Pfade abzubrechen und eine neue *Route Discovery* zu starten.

2. Der Sender lernt die Routen zu allen Knoten, die sich auf einer Route zu t befinden.

3. Das Source-Route-Paradigma, dass ein Nachrichtenpaket die im Header angegebene Route nimmt, wird beibehalten. Dies ist nicht der Fall, wenn ein Zwischenknoten entscheiden kann, ob ein Paket über eine alternative Route versendet wird.

4 Physische Parameter

In einem weiteren Teil der Arbeit wurden einige physische Parameter eines Ad-Hoc-Netzes auf WLAN-Basis gemessen. Für die Messungen wurden Laptops verwendet, die mit einer ad-hoc-fähigen Wireless LAN-Karte ausgerüstet waren. Durch die Zubehör-Software dieser Karten wird nur sehr rudimentäre Information über einige nicht näher spezifizierte Parameter wie Signalstärke und Signalqualität mitgeteilt. Das Ziel dieses Teils der Arbeit war, nähere Angaben zu folgenden Punkten machen zu können:

1. Maximale Übertragungsentfernung
2. Datenverlustrate
3. Auswirkung von Interferenzen

4.1 Messvorrichtungen

Da es keine Möglichkeit gibt, direkt auf die Schnittstelle der LAN-Karten zuzugreifen, wurden die Messungen mit einer Java-Applikation vorgenommen. Mit diesem Programm können Datenpakete verschiedener Grössen über eine Single-Hop-Verbindung gesendet werden. Die Applikation benutzt dabei ein Protokoll, das von der *Distributed Computing Group* für die Übungen der Vorlesung *Mobile Computing* entwickelt worden ist.

Die Messungen wurden allesamt in einem Gebäude (IFW) vorgenommen. Ein Laptop diente jeweils als Empfänger, einer als Sender. Der Sender sendete von verschiedenen Standorten aus Paketserien mit je 20 Datenpaketen à 10 Byte, 1 KByte und 64 KByte. Eine Messreihe wurde jeweils zweimal wiederholt.

Die Standorte, von denen aus gesendet wurde, waren so gewählt, dass bei allen Versuchen die Paketserien jeweils mit gleicher Sendestärke gesendet wurden. Zusätzlich wurde die Distanz zum Empfänger gemessen. In allen hier vorgestellten Messreihen wurde mit der maximalen Sendeleistung von 50 mW gesendet.

Um die Interferenzen zu messen, wurde ein dritter Laptop benutzt. Dieser diente als Störsender, der unmittelbar neben dem Empfänger unablässig Datenpakete versendete.

4.2 Resultate

In diesem Kapitel werden die Resultate der verschiedenen Messreihen vorgestellt. Die Ergebnisse sind in Abbildung 4 graphisch dargestellt. Für alle Versuche wurde die gleiche Darstellung gewählt: Die x-Achse zeigt die Signalstärke, mit der gesendet wurde, die y-Achse die Anzahl Datenpakete die empfangen wurde. Pro Sendeserie wurden jeweils 60 Pakete versendet.

4.2.1 Messungen ohne Interferenzen

In einer ersten Messreihe wurden nur zwei Laptops, ein Sender und ein Empfänger, benutzt. Es interessierte die maximale Reichweite sowie die Datenverlustrate. Zusätzlich wurde versucht, eine Verbindung zwischen der angegebenen Signalstärke und diesen Parametern herzustellen.

Diese Messungen wurden unter zwei verschiedenen Bedingungen durchgeführt: Im ersten Fall befand sich der Empfänger in einem offenen Raum, der Sender ausserhalb dieses Raumes. Im zweiten Fall waren Sender und Empfänger durch eine Betonmauer getrennt.

Senden durch einen offenen Raum: Wie man der Abbildung 4(a) entnehmen kann, nimmt die Datenempfangsrate nicht proportional zur Signalstärke ab. Bis zu einem gewissen Schwellwert werden die meisten Datenpakete empfangen, danach fast keine mehr. Dieser Schwellwert scheint spezifisch für die Paketgrösse zu sein und liegt bei kleinen Datenpaketen bei circa 15% Signalstärke. Grössere Datenpakete weisen das gleiche Verhalten auf, nur wird dieser Schwellwert früher erreicht.

Bei dieser Messreihe betrug die maximale Distanz, wo noch Datenpakete empfangen wurden, circa 19 m. Im Freien wurden allerdings deutlich längere Distanzen gemessen (bis circa 30 m). Im Allgemeinen scheint die Lage der Kommunikationspartner bezüglich Mauern, Türen, Gängen etc. einen erheblichen Einfluss auf die verschiedenen Parameter zu haben. Dies wird auch durch die zweite Messreihe bestätigt.

Senden durch eine Betonmauer: Auch in dieser Anordnung bestätigt sich obige Beobachtung: Die Datenempfangsrate ist bis zu einem gewissen Wert hoch, danach sehr tief. Vergleicht man Abbildung 4(b) mit Abbildung 4(a), so erkennt man, dass die Datenverlustraten bezüglich der Signalstärke fast gleich sind. Dies ist ein starkes Indiz dafür, dass eine Korrelation zwischen der Datenverlustrate und der Signalstärke besteht.

Die maximale Reichweite in dieser Messreihe beträgt 8 m. Die Mauer dient nicht nur als einfacher Dämpfer, die Signalstärke vermindert sich auch danach schneller als in der Messreihe oben. Allerdings muss darauf hingewiesen werden, dass die beiden Versuche nicht am genau gleichen Ort durchgeführt werden konnten. Es können also noch weitere bauliche Faktoren zu einer stärkeren Dämpfung geführt haben.

4.2.2 Messungen mit Interferenzen

In diesen Messreihen wurde der dritte Laptop unmittelbar neben den Empfänger gestellt. Dieser Laptop diente als Störsender und versendete fort-

laufend Datenpakete, wobei der Empfänger diese Datenpakete nicht zu empfangen hatte.

Kleine Daten-Pakete: Zuerst wurden vom Störsender 10 Byte grosse Datenpakete versendet. Abbildung 4(c) zeigt ein ähnliches Bild wie Abbildung 4(a), wo kein Störsender vorhanden ist, mit dem Unterschied, dass grosse Datenpakete von Anfang an nicht mehr empfangen werden. Es wird allerdings die gleiche maximale Reichweite erreicht wie ohne Störsender.

Diese Graphik zeigt eine einzelne Messreihe und nicht die Zusammenfassung mehrerer. Damit soll illustriert werden, dass zwischendurch ganze Paketserien den Empfänger nicht mehr erreichten. Dies sieht man bei der Serie, die bei 20% Signalstärke versendet wurde.

Grosse Daten-Pakete: Der gleiche Versuch, wobei der Störsender 10 KByte grosse Datenpakete versendet, zeigt, dass es zu *Verstopfungen* kommen kann. Die Verlustrate steigt sehr schnell an (Abbildung 4(d)) und erreicht 100%. Praktisch das gleiche Bild ergibt sich, wenn der Sender eine andere $SSID$ benutzt, d.h. sich logisch in einem anderen Netzwerk befindet (Abbildung 4(e)). Dieser Effekt ist also unabhängig davon, ob die Stationen sich im gleichen Netzwerk befinden.

Diese Messreihe bestätigt eine Beobachtung, die während den Versuchen gemacht wurde: An manchen Tagen waren die Datenverlustraten vormittags deutlich kleiner als nachmittags, wenn sich plötzlich viele Studenten mit ihren Laptops auf dem WLAN befanden.

4.3 Fazit

Die Aufgabe, die physischen Parameter eines Ad-Hoc Netzes zu messen, erwies sich als schwierig. Besondere Mühe bereitete, die verschiedenen Einflussfaktoren auf die Messergebnisse zu separieren. So arbeitete beispielsweise das Java-Programm anfänglich mit mehreren Threads, was zu Ergebnissen führte, die ich falsch interpretierte. Ein zweites Beispiel ist, dass ich anfangs immer von den gleichen Distanzen aus sendete. Da es aber nicht möglich war, die Tests immer am genau gleichen Ort durchzuführen, flossen die baulichen Eigenheiten auf unerwünschte Weise in die Ergebnisse mit ein. So macht es z.B. einen Unterschied, ob der Gang nach 7 m oder nach 10 m abbiegt. Nachdem ich das Augenmerk vermehrt auf die Signalstärke richtete, liessen sich Messergebnisse eher reproduzieren.

Die Hauptaussagen, die aus diesen Messreihen gemacht werden können, sind die folgenden:

1. Die Datenverlustrate nimmt nicht proportional mit der Signalstärke ab, sondern fällt bis zu einem spezifischen Wert schwach, danach rapide ab. Dies stellt ein Argument für eine Modellierung von Netz-

werkkarten mittels Sende- bzw. Empfangsradien dar, was schliesslich zu einer Representation durch Graphen führt.

2. Bauliche Eigenheiten der Umgebung haben einen grossen Einfluss auf den Empfangsradius.

3. Das Versenden von grossen Datenpaketen erhöht den Interferenzeffekt sehr stark.

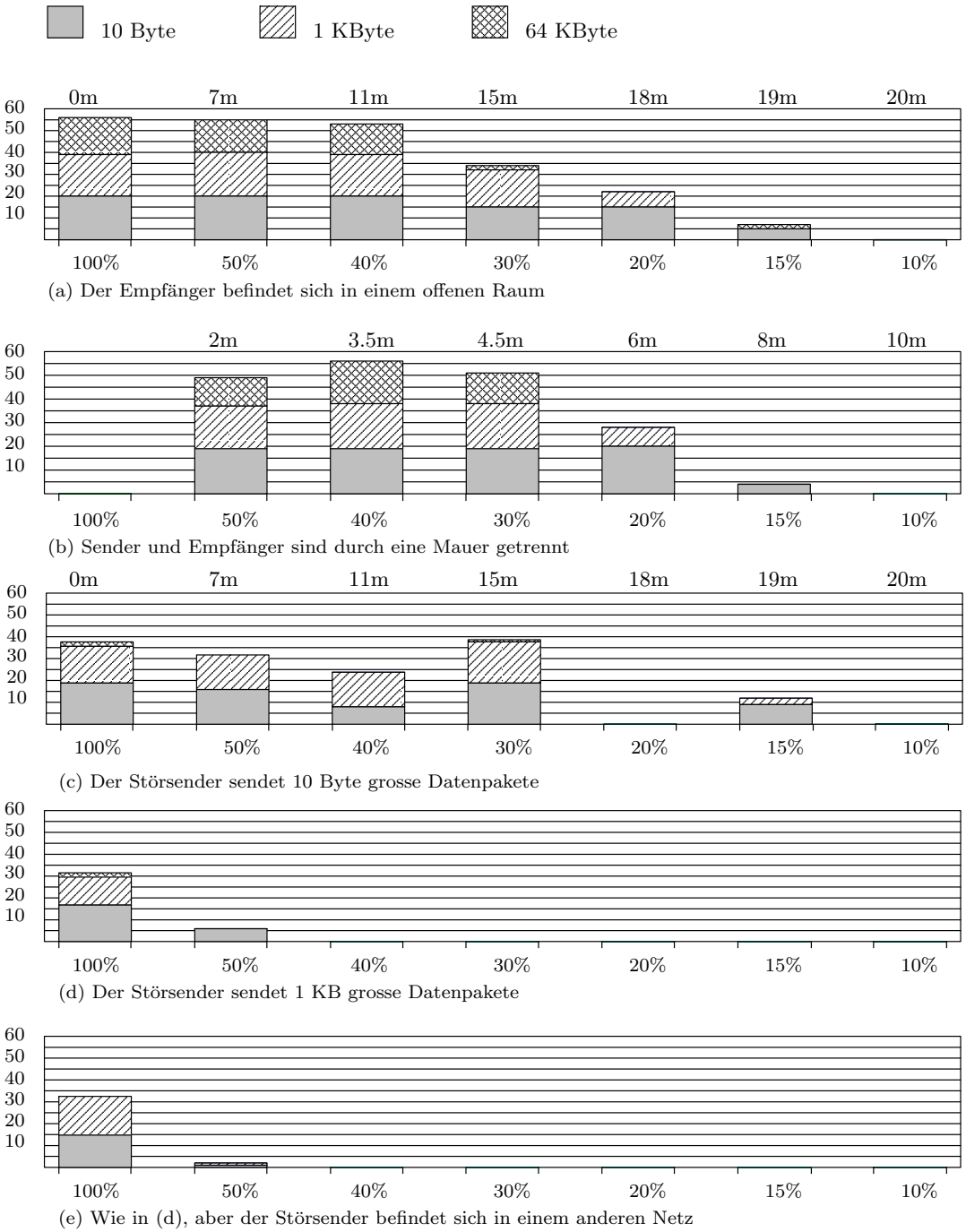


Abbildung 4: Schematische Darstellungen der verschiedenen Messreihen. Die y-Achse zeigt die Anzahl Pakete, die empfangen wurde, die x-Achse die Signalstärke, mit der gesendet wurde. Oberhalb der Schemata sind die Distanzen zwischen Sender und Empfänger aufgetragen.

Literatur

[1] Samir R. Das, Charles E. Perkins and Elizabeth M. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks.
http://www.iprg.nokia.com/~charliep/

[2] Elizabeth M. Royer and Chai-Keong Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks.
http://www.ces.clemson.edu/~rsass/courses/ NRG/Papers/Royer.pdf

[3] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols.
http://www.ics.uci.edu/~atm/adhoc/paper-collection/papers.html

[4] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad-Hoc Wireless Networks.
http://www.monarch.cs.rice.edu/papers.html

[5] David B. Johnson, David A. Maltz, Josh Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks.
http://www.monarch.cs.rice.edu/papers.html

[6] Yih-Chun Hu and David B. Johnson. Implicit Source Routes for On-Demand Ad Hoc Networking Routing.
http://www.monarch.cs.rice.edu/papers.html

[7] Jiři Matoušek and Jaroslav Nešetřil. Invitation to Discrete Mathematics, Oxford University Press, Oxford, 1998.