Optimierung von PRoPHET gegenüber bösartigen Knoten in opportunistischen Netzwerken

Bachelorarbeit

von

Andreas Lisowsky

aus

Langenfeld

vorgelegt am

Lehrstuhl für Technik Sozialer Netzwerke
Jun.-Prof. Dr.-Ing. Kalman Graffi
Heinrich-Heine-Universität Düsseldorf

März 2018

Betreuer:
Raphael Bialon, M. Sc.

Abstract

Opportunistische Netzwerke sind eine Art von Netzwerken, in welchen keine konstanten Verbindungen zwischen Knoten garantiert werden können. Sie sind eine Unterklasse von Delay/Disruption Tolerant Networks. In derartigen Netzwerken werden spezielle Routing-Protokolle benötigt, wie zum Beispiel PRoPHET, welches jeden Knoten eine Vorhersagbarkeitstabelle führen lässt. Diese Tabelle wird dafür verwendet, für jedes individuelle Paket den nächstbesten Hop zu bestimmen. Wie praktisch jedes Netzwerk sind auch opportunistische Netzwerke anfällig gegenüber Angriffen vielerlei Art. Das Ziel dieser Arbeit war es das Routing-Protokoll PRoPHET innerhalb des Simulators PeerfactSim.KOM so zu erweitern, dass eine gewisse Teilmenge von Angriffen, die im Verlauf dieser Arbeit vorgestellt werden, unterbunden werden kann. Die Angriffe, welche betrachtet wurden, haben bösartige Knoten in das Netzwerk integriert, welche entweder die time to live eines Pakets auf einen geringen Wert setzen oder falsche Einträge in ihrer Vorhersagbarkeitstabelle verwenden. Beide Methoden der Angreifer sollten unterbunden werden, indem die normalen Knoten untersuchen, ob eine eingehende time to live eines Pakets oder ein Tabelleneintrag eines Nachbarknotens valide Werte beinhalten. Ob ein Wert valide ist, wurde mit gewissen Formeln und Methoden definiert. Um die Auswirkung der bösartigen Knoten und den Einfluss der Gegenmaßnahmen zu messen, wurden die Latenz, die Anzahl der Hops und die Zustellungsrate in den jeweiligen Angriffsszenarien betrachtet. Die Metriken wurden durch die bösartigen Knoten stark verschlechtert. Die vorgestellte Gegenmaßnahme für den Angriff, welcher die time to live manipuliert, hat die Auswirkung der bösartigen Knoten gut unterbinden können, die Gegenmaßnahme für die manipulierten Vorhersagbarkeitstabellen jedoch nicht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In der heutigen Zeit zeichnen sich die meisten Netzwerke dadurch aus, dass sie konstante Verbindungsstrecken zwischen allen Knoten besitzen, sich also immer eine Route zur Datenübertragung finden lässt. Es gibt allerdings auch sogenannte DTNs (**D**elay/**D**isruption **T**olerant **N**etworks), welche keine garantierten Verbindungsstrecken zwischen allen Knoten besitzen. In [RCYC10] erklären die Autoren die generelle Funktionsweise von DTNs.
Ein Spezialfall von den DTNs sind die sogenannten opportunistischen Netzwerke (siehe [CGMP10]), in welchen die Knoten sich nur in bestimmten Zeitintervallen (oder gar nicht) zueinander in Reichweite befinden. Für solche Netzwerke existieren zahlreiche Routing-Protokolle.
Ein Protokoll welches häufig für DTNs verwendet wird ist PRoPHET, (siehe [LDDG12]) in welchem jeder Knoten in einer Vorhersagbarkeitstabelle einträgt wie wahrscheinlich es ist Daten erfolgreich zu einem Zielknoten zu routen, wobei diese Tabelle häufig aktualisiert wird. Wenn zwei Knoten sich zueinander in Verbindungsreichweite bewegen, tauschen sie ihre Tabellenwerte aus und berechnen mithilfe dieser zusätzlichen Informationen neue Werte für bestimmte Tabelleneinträge. Eine Datenübertragung muss nicht immer direkt zum Zielknoten erfolgen, da es gegebenenfalls besser ist die Daten über eine Knotenkette zu leiten. PRoPHET funktioniert also nur dann reibungslos, wenn alle Netzwerkknoten gutwillig sind und korrekte Vorhersagbarkeitseinträge propagieren.
In Kapitel 2 werden verwandte Arbeiten vorgestellt, in welchen verschiedene Angriffe auf DTNs untersucht werden. Danach werden in Kapitel 3 die für diese Arbeit relevanten Grundlagen im Detail erklärt. Kapitel 4 bietet eine Übersicht über eine Vielzahl von möglichen Angriffen und deren Relevanz bezüglich dieser Arbeit. Kapitel 5 behandelt die Simulations-

ergebnisse bezüglich des unmodifizierten PRoPHETs. Danach werden in Kapitel 6 einzelne Modifikationen von PRoPHET im Detail erklärt. Was die Auswirkungen der einzelnen PRoPHET Modifikationen auf die Simulationsergebnisse sind, wird in Kapitel 7 beschrieben. Anschließend werden in Kapitel 8 noch einmal die Ergebnisse dieser Arbeit zusammengefasst und ein Ausblick auf die Zukunft geboten, in welchem beschrieben wird an welchen Teilen der Arbeit noch weiter gearbeitet werden könnte.

Kapitel 2

Verwandte Arbeiten

In [BG16] haben die Autoren untersucht, welche Auswirkungen bösartige Knoten auf Netzwerkmetriken wie Zustellungsrate, Verzögerung und Sendungsanzahl von Paketen haben. Es wurde eine Vielzahl von Angriffsmöglichkeiten verwendet, welche in die Kategorien "No data routing", "Modification of routing information" und "Overloading other nodes" unterteilt wurden. Es wurden allerdings keine Gegenmaßnahmen für die Angriffe implementiert oder getestet. In dieser Arbeit wird ein Teil von diesen möglichen Angriffen betrachtet und versucht Gegenmaßnahmen für diese zu entwickeln.
In [NAP14] haben sich die Autoren mit einer Vielzahl von flooding Angriffen auf verschiedenste Routing-Protokolle (darunter auch PRoPHET) für DTNs beschäftigt. Es wurde betrachtet, welchen Einfluss die Größe des Puffers der einzelnen Knoten auf Netzwerkmetriken wie Nachrichtenzustellungsrate, Größe des Nachrichtenoverheads, durchschnittliche Latenz und Anzahl der verworfenen Nachrichten hat. Was flooding Angriffe sind, wird in Kapitel 4 beschrieben. Eine Vergrößerung des Puffers auf einen Wert von bis zu 4900% hat gewaltige positive Auswirkungen auf diese Metriken gehabt.
Der Lösungsansatz für die verschiedenen Angriffe ist für diese Arbeit nicht so sehr von Bedeutung, da der Puffer eine knotenabhängige Komponente ist und in dieser Arbeit nur betrachtet wird wie PRoPHET modifiziert werden kann. Nichtsdestotrotz ist die Arbeit jedoch relevant, da in ihr eine Vielzahl von verschiedenen Angriffsmöglichkeiten für einen flooding Angriff vorgestellt werden.
In der Arbeit [RCYC10] geht es um sogenannte Wormhole Angriffe in DTNs. Sie beschäftigt sich speziell mit dem Erkennen von Wormhole Angriffen in DTNs. Was Wormhole Angriffe sind wird in Kapitel 4 beschrieben. Das Problem in DTNs bezüglich der Erkennung von

Wormhole Angriffen ist, dass es keine zentrale Verwaltungsinstanz für die Knoten des Netzwerks gibt, so dass schwerer erkannt werden kann, ob eine bestimmte Datenübertragung legitim war. Um nun auch in einem DTN einen solchen Angriff erkennen zu können, haben die Autoren ausgenutzt, dass durch das Tunneln von Daten bestimmte Distanzverhältnisse in der Netzwerktopologie erzeugt werden, welche unter normalen Umständen nicht auftreten könnten. Wird ein solches unmögliches Verhältnis erkannt, geht das System davon aus, dass es sich um einen Angriff handeln muss. In der Arbeit wurde allerdings nur das Erkennen eines Wormhole Angriffs behandelt. Um den Angriff zu unterbinden, müssten jedoch alle Knoten, die Teil des Wormhole Angriffs sind, aus dem Netzwerk entfernen werden. Der Lösungsansatz könnte also noch erweitert werden.
Es wurden also in diesen drei Arbeiten viele Angriffe vorgestellt, meistens aber keine Gegenmaßnahmen entwickelt. Wurde doch eine Gegenmaßnahme entwickelt, dann war diese nicht Teil des Routing-Protokolls. In dieser Arbeit wird betrachtet wie PRoPHET modifiziert werden kann um Angriffe zu unterbinden.

Kapitel 3

Grundlagen

3.1 DTN (Delay/Disruption Tolerant Network)

DTNs werden in [RCYC10] beschrieben. Als DTN wird im Allgemeinen ein Netzwerk bezeichnet, in welchem keine Garantie für die Existenz einer vollständigen Verbindungsstrecke zwischen Knoten besteht. In anderen Worten, die Verbindungsstrecke kann unterbrochen (disrupted) werden. Da das Netzwerk keine konstanten Verbindungsstrecken besitzt, können also auch keine Routing-Protokolle verwendet werden, die dies voraussetzen. In Netzwerken mit konstanten Verbindungsstrecken könnte das Routing realisiert werden, indem jeder Knoten eine feste Liste von Hops besitzt, an welche ein Paket weitergeleitet wird. In DTNs werden spezielle Routing-Protokolle benötigt, welche darauf ausgelegt sind in einem sich ständig verändernden Netzwerk Pakete bestmöglichst ans Ziel zu bringen. PRoPHET ist eins dieser Protokolle.
Weil DTNs nicht über festbestehende Verbindungsstrecken leiten, kann die Dauer der Übertragung von Daten stark variieren, je nachdem wie schnell eine effiziente Route von Knoten zu Knoten zum Ziel gefunden wird. Das Netzwerk muss also Verzögerungen (delay) tolerieren.

3.2 Opportunistisches Netzwerk

Opportunistische Netzwerke sind ein Spezialfall von DTNs und es kann in [CGMP10] nachgelesen werden, was die Motivation hinter ihnen ist oder in [DL09], welche Sicherheitsprobleme sich mit ihnen ergeben.
In einem opportunistischen Netzwerk sind die einzelnen Knoten mobile Geräte, welche sich in einem gewissen Umfeld bewegen und eine geringe Verbindungsreichweite zueinander besitzen. In der Praxis also zum Beispiel Studenten, die mit ihren Handys auf dem Campus umherlaufen.
Da die Knoten sich ständig bewegen und nur eine geringe Verbindungsreichweite zueinander besitzen, existieren keine konstanten Verbindungspfade zwischen den Knoten. Ein solches Netzwerk muss also tolerant gegenüber Verzögerungen und Verbindungsabbrüchen sein, was es als ein DTN klassifiziert.
Solch ein Netzwerk ist potenziell eine sehr interessante Alternative zu fest bestehenden Netzwerken, da ein opportunistisches Netzwerk ohne zentrale Verwaltungsinstanz funktioniert.

3.3 PRoPHET

PRoPHET (**P**robabilistic **Ro**uting **P**rotocol using **H**istory of **E**ncounters and **T**ransitivity) ist ein Routingprotokoll für DTNs, welches jeden Knoten ständig eine Tabelle aktualisieren lässt, welche die Vorhersagbarkeit für erfolgreiches Routen zu einem Zielknoten für alle Knotenpaare beinhaltet. PRoPHET ist sehr detailliert in [LDDG12] beschrieben. Es wird stets versucht anhand dieser Tabelle zu entscheiden, ob es besser ist ein Paket direkt ans Ziel zu übertragen oder ob es an einen Nachbarknoten abgegeben werden soll. Wenn zwei Knoten sich treffen, durchlaufen diese immer zwei Phasen. In der ersten Phase werden die Tabelleneinträge aktualisiert. In der zweiten Phase tauschen sich die Knoten über die zu verteilenden Pakete aus und verwenden zusätzlich die aktualisierten Tabelleneinträge, um zu entscheiden wie welches Paket weitergeleitet werden soll.
Falls nicht anders erwähnt, sei jetzt immer Knoten A der momentan betrachtete Knoten, Knoten B ein Knoten in Reichweite von A und Knoten C ein beliebiger anderer Knoten.

3.3.1 Aktualisieren der Vorhersagbarkeitstabelle

Die Werte in der Tabelle werden nun wie folgt bestimmt. Die Vorhersagbarkeit einer erfolgreichen Datenzustellung von Knoten A nach Knoten B wird mit $P(A,B)$ bezeichnet. Falls ein Knoten A für $P(A,B)$ in seiner Tabelle noch keinen Eintrag besitzt, wird $P(A,B)$ als 0 betrachtet. Falls ein Knoten A einem Knoten B begegnet und der Knoten A noch keinen Eintrag für $P(A,B)$ besitzt, wird er auf einen vorher fest definierten Startwert $P_{encounter_{first}}$ gesetzt, welcher im Regelfall 0,5 ist. Ein Eintrag kann fehlen, wenn die Knoten sich zum ersten Mal treffen oder wenn sie sich seit langer Zeit nicht mehr getroffen haben.
Wenn zwei Knoten A und B sich begegnen, aktualisieren diese Knoten mithilfe der jeweils anderen Tabelle ihre Einträge. Es wird zum einen die Vorhersagbarkeit der direkten Übertragung von A nach B durch Gleichung 3.1 aktualisiert. Danach werden alle anderen Vorhersagbarkeiten durch Gleichung 3.2 aktualisiert, (B ist in der Gleichung also ein beliebiger Knoten, nicht mehr der einzelne Knoten, den A gerade getroffen hat) damit ältere Einträge ständig reduziert werden. Dann wird noch zusätzlich die Vorhersagbarkeit der indirekten Übertragung von A über B nach C, wie in Gleichung 3.3 aktualisiert. Die Eigenschaft die durch Gleichung 3.3 realisiert wird, wird auch auch als Transitivität bezeichnet.

$$P(A,B) = P(A,B)_{old} + (1 - \delta - P(A,B)_{old}) \cdot P_{encounter}(intvl) \tag{3.1}$$

$$P(A,B) = P(A,B)_{old} \cdot \gamma^k \tag{3.2}$$

$$P(A,C) = \max\{P(A,C)_{old}, P(A,B) \cdot P(B,C)_{received} \cdot \beta\} \tag{3.3}$$

In Gleichung 3.1 ist δ eine sehr klein gewählte Konstante, zum Beispiel 0,01, deren einzige Funktion es ist dafür zu sorgen, dass $P(A,B) < 1$ gilt. Der Wert 1 ist nämlich für den Spezialfall $P(A,A)$ reserviert. Es gilt $0 \leq P_{encounter} \leq 1$ und ist eine Funktion, welche bestimmt wie schnell $P(A,B)$ gegen 1 konvergiert. $P_{encounter}$ ist gegeben durch die Gleichung 3.4.

$$P_{encounter}(intvl) = \begin{cases} P_{encounter_{max}} \cdot \frac{intvl}{I_{typ}} & \text{for } 0 \leq intvl \leq I_{typ} \\ P_{encounter_{max}} & \text{for } intvl > I_{typ} \end{cases} \tag{3.4}$$

$P_{encounter_{max}}$ ist der Maximalwert für $P_{encounter}$ und sollte relativ hoch gewählt werden, zum Beispiel 0,7, damit der Wert für $P(A,B)$ schneller wächst als er durch die Gleichung 3.2 altert. Die Variable $intvl$ ist die Größe des Zeitintervalls, vom letzten Treffen der Knoten A und B bis zum momentanen Treffen. Die Konstante I_{typ} ist die typische Intervallgröße. Wenn $intvl < I_{typ}$ wollen wir den Wert von $P_{encounter}$ verringern, damit kurze Verbindungsabbrüche

weniger Auswirkungen auf die Tabellenaktualisierungen haben. Kurze Verbindungsabbrüche würden ansonsten PRoPHET fälschlich annehmen lassen, dass sich die beiden Knoten erneut getroffen hätten.
In Gleichung 3.2 ist γ eine Konstante, welche etwas kleiner als 1 ist, zum Beispiel 0,99. Dadurch wird mithilfe des Exponenten k der Wert von $P(A,B)$ langsam reduziert. Die Variable k gibt an, wie viele Zeitintervalle von fester, vordefinierter Länge seit dem letzten Aktualisieren von $P(A,B)$ verstrichen sind. Wenn durch das Reduzieren eines Eintrags $P(A,B)$ der Wert unter $P_{first_{threshold}}$ sinkt, wird beim nächsten Treffen von A und B in Gleichung 3.1 der Wert von P(A,B) wieder auf $P_{encounter_{first}}$ gesetzt. $P_{first_{threshold}}$ ist üblicherweise 0,1.
In Gleichung 3.3 ist $P(B,C)_{received}$ ein Eintrag, den Knoten A von Knoten B (nach Anwenden von Gleichung 3.2) erhält. β ist eine Konstante und es gilt $0 \leq \beta \leq 1$, wobei häufig 0,9 verwendet wird.
In Tabelle 3.1 sind noch einmal, der Übersicht wegen, alle Konstanten und deren Standardwerte angegeben. Diese Standardwerte müssen nicht zwangsweise verwendet werden, die Konstanten können auch für ein spezifisches Netzwerk angepasst werden. Allerdings sind die Standardwerte so gewählt, dass mit ihnen ein brauchbares Routing in fast allen Netzwerken erreicht wird.

Tabelle 3.1: Konstanten und deren Standardwerte

Konstante	Standardwert
$P_{encounter_{max}}$	0,7
$P_{encounter_{first}}$	0,5
$P_{first_{threshold}}$	0,1
β	0,9
γ	0,999
δ	0,01
I_{typ}	Abhängig vom Netzwerk

3.3.2 Routing/Forwarding

In Routing-Protokollen mit festbestehenden Verbindungspfaden ist die Entscheidungsfindung für Routen relativ einfach, es wird der Pfad mit den geringsten Kosten verwendet. Das Problem besteht also nur in der Berechnung der Kosten.

In PRoPHET ist das Ganze weit komplizierter, da kein Pfad eine Übertragung zum Ziel garantiert. Damit also dafür gesorgt ist, dass die Wahrscheinlichkeit für ein Paket sein Ziel zu erreichen relativ hoch ist, müssen Pakete mehrfach über mehrere Routen geschicket werden. In diesem Szenario muss also abgewägt werden, ob mehr Pakete gesendet werden und dadurch das Netzwerk stärker ausgelastet wird, dafür aber mehr erfolgreiche Übertragungen erzielt werden, oder ob eine geringe Anzahl von Paketen gesendet und dadurch das Netzwerk weniger ausgelastet wird, dafür aber weniger erfolgreiche Übertragungen erzielt werden.
Es können jetzt gewisse Regeln in PRoPHET definiert werden, um zu entscheiden, wie welche Pakete weitergeleitet werden sollen. Es seien Knoten A und B zwei Knoten, die sich gerade begegnen. Es sei D ein Zielknoten und für beliebige Knoten X und Y sei $P(X,Y)$ der Eintrag in X für die Vorhersagbarkeit einer erfolgreichen Übertragung nach Y. Wenn wir von A aus weiterleiten wollen, dann könnte zum Beispiel überprüft werden ob $P(B,D) > P(A,D)$ und falls dies gilt über B weiterleiten. Es könnte auch die maximale Anzahl der Kopien eines Pakets, die ein Knoten verschicken darf, limitert werden und noch vieles mehr. In [LDDG12] wird eine Vielzahl von möglichen Regeln für das Weiterleiten von Paketen präsentiert.
In Abbildung 3.1 ist in Pseudo-Code angegeben, wie das Forwarding in PeerfactSim.KOM

```
if packet hop count < TTL then
    for all neighbors N do
        if packet was not sent by N and N has no copy of it already then
            if N is receiver of packet then
                send packet to N;
            else
                if P(N,D) > P(A,D) then
                    send packet to N for further forwarding;
                end
            end
        end
    end
else
    remove packet from buffer (drop it);
end
```

Abbildung 3.1: Algorithmus zum Forwarding in PeerfactSim.KOM

(siehe [Gra11]) implementiert ist. Es seien nun A und B Knoten, die sich gerade getroffen haben und die Forwardingphase durchlaufen. A entscheidet nun was mit den Paketen aus dem Puffer geschehen soll. Es sei D immer der Zielknoten eines Pakets.
Es wurde also geschildert, was DTNs und opportunistische Netzwerke sind und wie PRoPHET funktioniert. Mithilfe der genauen Spezifikation von PRoPHET können nun im Detail

Angriffe und Gegenmaßnahmen für diese betrachtet werden.

Kapitel 4

Attack Tree

Es wird nun ein Attack Tree erstellt, welcher eine Vielzahl potenzieller Angriffe beinhaltet. Die Angriffe, für welche es sich nicht lohnt Gegenmaßnahmen zu entwickeln, werden herausgefiltert. Wenn es sich nicht lohnt, wird der Angriff als irrelevant bezeichnet, ansonsten als relevant. Ob ein Angriff relevant ist, hängt von mehreren Eigenschaften ab. Es werden die Kosten für den Angreifer, die Möglichkeit der Existenz einer Lösung und das Ausmaß der Auswirkungen des Angriffs auf das Netzwerk berücksichtigt. In Tabelle 4.1 ist der Attack Tree dargestellt.

Tabelle 4.1: Attack Tree

4.1.1 Black Hole
4.1.1a TTL Min
4.1.1b Alle Pakete verwerfen
4.1.1c Falsche Vorhersagbarkeitstabelle + TTL Min
4.1.2 Topologie Störung
4.1.2a Wormhole Angriff
4.1.3 Flooding
4.1.3a Packet Flood Angriff
4.1.3b Replica Flood Angriff
4.1.3c Selective Flooding

4.1 Die Einzelnen Angriffe im Detail

4.1.1 Black Hole

Als ein Black Hole Angriff werden alle Angriffe bezeichnet, bei denen ein bösartiger Knoten versucht so viele Daten von Nachbarknoten entgegenzunehmen wie möglich, um dann dafür zu sorgen, dass diese nicht ans Ziel gelangen. In [BG16] haben die Autoren einige Black Hole Angriffe vorgestellt, von denen jetzt einige in den Attack Tree übernommen werden.

4.1.1a TTL Min

Bei diesem Angriff setzt ein bösartiger Knoten den TTL (Time To Live) Wert eines Pakets auf einen geringen Wert, wodurch deutlich weniger Pakete ihr Ziel erreichen können, da die potenzielle Anzahl der Hops entlang des Übertragungsweges drastisch reduziert wird.
Dieser Angriff kostet den Angreifer fast keine Ressourcen und ist einfach zu implementieren.
Bewertung: relevant

4.1.1b Alle Pakete verwerfen

Bei diesem Angriff verwirft ein bösartiger Knoten alle eingehenden Pakete. Dieser Angriff kostet den Angreifer fast keine Ressourcen und kann verheerende Auswirkungen auf die Qualität des Netzwerks haben, abhängig davon, wie oft Pakete über den bösartigen Knoten geleitet werden.
Bewertung: relevant

4.1.1c Falsche Vorhersagbarkeitstabelle + TTL Min

Dieser Angriff ist sehr ähnlich zu dem aus 4.1.1a, mit dem Unterschied, dass zusätzlich auch noch falsche Vorhersagbarkeitstabellenwerte vom bösartigen Knoten propagiert werden. Der bösartige Knoten trägt in seiner gesamten Tabelle Werte ein, die sehr hoch sind, und somit

erscheint der bösartige Knoten als bester Kandidat zum Weiterleiten von Paketen. 4.1.1c ist also praktisch der worst case von 4.1.1a.
Die falschen Einträge in der Vorhersagbarkeitstabelle sorgen allerdings nicht nur dafür, dass der bösartige Knoten mehr Pakete verwerfen kann, sondern auch dafür, dass sämtliche Tabelleneinträge im Netzwerk sich langsam verfälschen. Der bösartige Knoten hat also auch eine stark infektiöse Charakteristik.
Bewertung: relevant

4.1.2 Topologie Störung

Viele Routing-Algorithmen für DTNs basieren auf dem Erstellen einer gewissen Netzwerktopologie durch das Führen von Daten, welche diese entweder direkt oder indirekt beschreiben. PRoPHET lässt jeden Knoten eine Vorhersagbarkeitstabelle führen, so dass ein Knoten A, der häufig in Reichweite zu Knoten B, ist hohe Wahrscheinlichkeitswerte für den Eintrag $P(A,B)$ in seiner Tabelle hat. Deswegen lässt sich annehmen, dass alle Knotenpaare $\{A,B\}$ sich im Netzwerk nah zueinander befinden, falls die Einträge $P(A,B)$ und $P(B,A)$ hoch sind. PRoPHET führt also somit keine Daten, die die Netzwerktopologie direkt beschreiben, aber immerhin indirekt.
Es gibt Angriffe, welche versuchen die topologische Ansicht der Knoten bezüglich des Netzwerks zu stören, indem sie in irgendeiner Weise die geführten Daten bezüglich der Netzwerktopologie manipulieren. Das Manipulieren kann direkt oder indirekt geschehen, wobei das direkte Ändern deutlich schwieriger zu erreichen ist, da dafür Zugriff auf die gutartigen Knoten benötigt werden würde. Ein Beispiel für das indirekte Ändern der Daten wäre ein Wormhole Angriff, wie in [RCYC10] beschrieben.

4.1.2a Wormhole Angriff

Bei einem Wormhole Angriff versucht ein Angreifer das Netzwerk zu stören, indem zwei bösartige Knoten in das Netzwerk integriert werden, welche eine große Distanz zueinander haben und mithilfe eines Tunnels miteinander verbunden sind. Das Netzwerk soll glauben, dass diese beiden Angreiferknoten eine geringe Distanz zueinander haben, damit die anderen

Knoten im Netzwerk eine falsche topologische Ansicht des Netzwerks generieren.
Speziell PRoPHET ist leicht durch diesen Angriff beeinflussbar, da ein getunneltes Wormhole-Knotenpaar dafür sorgen würde, dass alle Knoten, die in Kontakt mit einem der beiden Angreiferknoten kommen, ihre Vorhersagbarkeitstabellen mit falschen Werten füllen.
Es ist extrem schwer für PRoPHET, oder für DTNs im Allgemeinen, einen Lösungsansatz für Wormhole Angriffe zu entwickeln, weil dafür eine zuverlässige Erkennungsmaßnahme benötigt werden würde, was aber ohne zentrale Verwaltungsinstanz nicht einfach ist. In [RCYC10] haben die Autoren eine Methode implementiert, welche recht zuverlässig Wormhole Angriffe erkennen kann, indem bestimmte Distanzverhältnisse im Netzwerk, die unter normalen Umständen nicht auftreten können, als Angriffe identifiziert werden.
Wenn die Auswirkungen eines Wormhole Angriffs betrachtet werden fällt auf, dass diese Art von Angriff ähnliche Auswirkungen auf PRoPHET hat, wie der Angriff in 4.1.1c. Es werden nämlich die Vorhersagbarkeitstabellenwerte der Knoten verfälscht. Der große Unterschied ist jedoch, dass ein Wormhole Angriff dafür sorgt, dass PRoPHET selbst die falschen Einträge verursacht, da alle Tabelleneinträge korrekt gemäß der Berechnungsformeln für die vorgetäuschte Netzwerktopologie aktualisiert werden. Bei Angriff 4.1.1c trägt der Angreifer unnatürliche, womöglich konstante, hohe Werte manuell in die Tabellen ein, welche sich dann infektiös im Netzwerk verbreiten. Das ist viel leichter zu bemerken, da die Tabelleneinträge höchstwahrscheinlich stark von den natürlich generierten Einträgen abweichen.
Die Kosten für diesen Angriff sind für den Angreifer relativ gering, die Auswirkungen verheerend, allerdings ist das Implementieren einer Lösung sehr umfangreich.
Bewertung: relevant

4.1.3 Flooding

In [NAP14] wurden mehrere flooding Angriffe beschrieben und untersucht, von denen jetzt einige in den Attack Tree übernommen werden. Ein Angriff wird als flooding Angriff bezeichnet, wenn ein bösartiger Knoten das Netzwerk stört, indem er eine Vielzahl von nutzlosen Paketen dauerhaft an alle Nachbarn sendet und somit das Verarbeiten von Paketen von nicht bösartigen Knoten verlangsamt oder sogar ganz verhindert. Wenn der Puffer eines Knotens vollständig mit nutzlosen Paketen des Angreifer gefüllt ist, kann dieser somit keine echten Pakete mehr annehmen. Dieser Angriff kann einen starken Einfluss auf die Netzwerkqualität haben, kostet den Angreifer allerdings viele Ressourcen, wie zum Beispiel Bandbreite.

Eine einfach Lösung um flooding Angriffe zu unterbinden wäre es, wie in [NAP14] angegeben, die Puffer der einzelnen Knoten zu vergrößern. Das wäre aber keine Modifikation von PRoPHET, sondern eine der Knoten selbst, was also für diese Arbeit uninteressant ist. Außerdem sorgt das Vergrößern des Puffers dafür, dass die Übertragungsdauer der einzelnen Pakete sich erhöht, diese Lösung hat also auch Nachteile und ist dementsprechend nicht optimal. Eine Lösung in PRoPHET für diesen Angriff wäre es, bösartige Knoten zu identifizieren und deren Pakete zu verwerfen, was den Puffer für die echten Pakete frei halten würde. Es wäre allerdings relativ schwer, einen Knoten als bösartig zu identifizieren, wenn er legitime Pakete verschickt. Zusätzlich, selbst wenn solche bösartigen Knoten mit perfekter Genauigkeit identifiziert werden könnten, wäre die Lösung trotzdem nur suboptimal. Denn der Knoten, der die bösartigen Pakete verwirft, könnte trotzdem durch den Empfang einer Vielzahl solcher Pakete überlastet werden, falls das reine Identifizieren und Verwerfen bereits die gesamte Bandbreite des Knotens verbraucht. Die Aussichten für das Finden einer zufriedenstellenden Lösung für flooding Angriffe in PRoPHET wäre also relativ schlecht, weswegen alle nun folgenden spezifischen flooding Angriffe als irrelevant bezeichnen werden.

4.1.3a Packet Flood Angriff

Der bösartige Knoten dupliziert beliebige Nachrichten, die er bereits von anderen Knoten erhalten hat und schickt diese an beliebige Nachbarn um diese zu überlasten. Das Duplizieren ist besonders einfach für den Angreifer zu implementieren, da er sich dann noch nicht einmal Gedanken darüber machen muss, wie er ein selbsterstelltes falsches Paket gestalten müsste.
Bewertung: irrelevant

4.1.3b Replica Flood Angriff

Der bösartige Knoten hat bei diesem Angriff nicht als Primärziel, die anderen Knoten zu behindern, sondern die eigenen Daten schneller im Netzwerk zu verbreiten. Der bösartige Knoten versucht dies zu erreichen, indem er alle Nachrichten die ihm selbst gehören mehr-

fach losschickt. Durch das Senden dieser erhöhten Anzahl an Daten verbraucht der bösartige Knoten mehr Netzwerkressourcen der anderen Knoten als ein gutartiger Knoten. Dieser Angriff ist für den Angreifer leicht zu implementieren, kostet ihn aber Bandbreite, wie alle flooding Angriffe.
Bewertung: irrelevant

4.1.3c Selective Flooding

Diese Art von flooding Angriff ist speziell für Routing-Algorithmen, welche ihre Daten über berechnete Pfade schicken, wozu auch PRoPHET gehört. Der bösartige Knoten entwirft Pakete, welche eine hohe Wahrscheinlichkeit haben weitergeleitet zu werden, da dadurch mehr falsche Nachrichten in Umlauf geraten und das Netzwerk belasten. Dafür schaut sich der bösartige Knoten einfach die Vorhersagbarkeitkeitswerte in den Tabellen eines Nachbarknotens an und schickt Pakete los, die als Ziel den Knoten haben, an welche der Nachbarknoten mit größter Wahrscheinlichkeit erfolgreich sendet. Dieser Angriff ist relativ leicht für den Angreifer zu implementieren.
Bewertung: irrelevant

Der neue Attack Tree, welcher nur noch die relevanten Angriffe beinhaltet, sieht nun wie in Tabelle 4.2 dargestellt aus.

Tabelle 4.2: Attack Tree (gefiltert nach Relevanz)

4.1.1 Black Hole
4.1.1a TTL Min
4.1.1b Alle Pakete verwerfen
4.1.1c Falsche Vorhersagbarkeitstabelle + TTL Min
4.1.2 Topologie Störung
4.1.2a Wormhole Angriff

Da diese Arbeit nur einen gewissen Umfang erlaubt, wird der Attack Tree noch weiter reduziert, welcher nun wie in Tabelle 4.3 dargestellt aussieht. Es werden diese zwei Angriffe ausgewählt, da diese sich sehr gut im Simulator implementieren lassen.

Tabelle 4.3: Attack Tree (reduziert)

4.1.1 Black Hole
4.1.1a TTL Min
4.1.1c Falsche Vorhersagbarkeitstabelle + TTL Min

Kapitel 5

Simulation mit unmodifiziertem PRoPHET

Es wird nun ein spezifisches Szenario simuliert und grafisch dargestellt welchen Einfluss die ausgewählten Angriffsarten aus Kapitel 4 auf die Latenz, die Anzahl der benötigten Hops und die Rate einer erfolgreichen Datenzustellung haben. Es werden immer die Durchschnittswerte über alle Knoten betrachtet.
Das Szenario sieht wie folgt aus: Es bewegen sich 100 Knoten auf einer 500x500 Meter großen Fläche randomisiert umher und verschicken im Minutentakt Ping-Nachrichten an sich gegenseitig. Die Simulationsdauer beträgt zwei Stunden. Die Puffergröße in den einzelnen Knoten ist so gewählt, dass genau 1000 Pakete in ihn hineinpassen, so dass möglichst wenig Pakete verworfen werden. Das soll dafür sorgen, dass die gemessenen Metriken ohne bösartige Knoten einen sehr guten Wert erreichen, damit anschließend die Verschlechterung durch die bösartigen Knoten besser sichtbar ist.
Damit allerdings brauchbare Simulationsergebnisse erzielt werden können, müssen auch noch die Konstanten der PRoPHET-Implementation sinnvoll gesetzt werden. In Kapitel 3 sind in Tabelle 3.1 die Parameter aufgelistet, welche berücksichtigt werden müssen. Alle diese Konstanten, außer I_{typ}, besitzen Standardwerte, welche für dieses Simulationsszenario auch passend sind und brauchbare Ergebnisse liefern. Die Variable k aus Gleichung 3.2 ist für dieses Simulationsszenario die Anzahl der vergangenen Minuten, seit der letzten Begegnung von zwei Knoten. Die Konstante I_{typ} ist stark abhängig von der Netzwerkkonfiguration, idealerweise sollte sie genau dem Zeitintervall entsprechen, in welchem sich im Durchschnitt ein beliebiges Knotenpaar genau einmal trifft. Die durchschnittliche benötigte Zeit für das Tref-

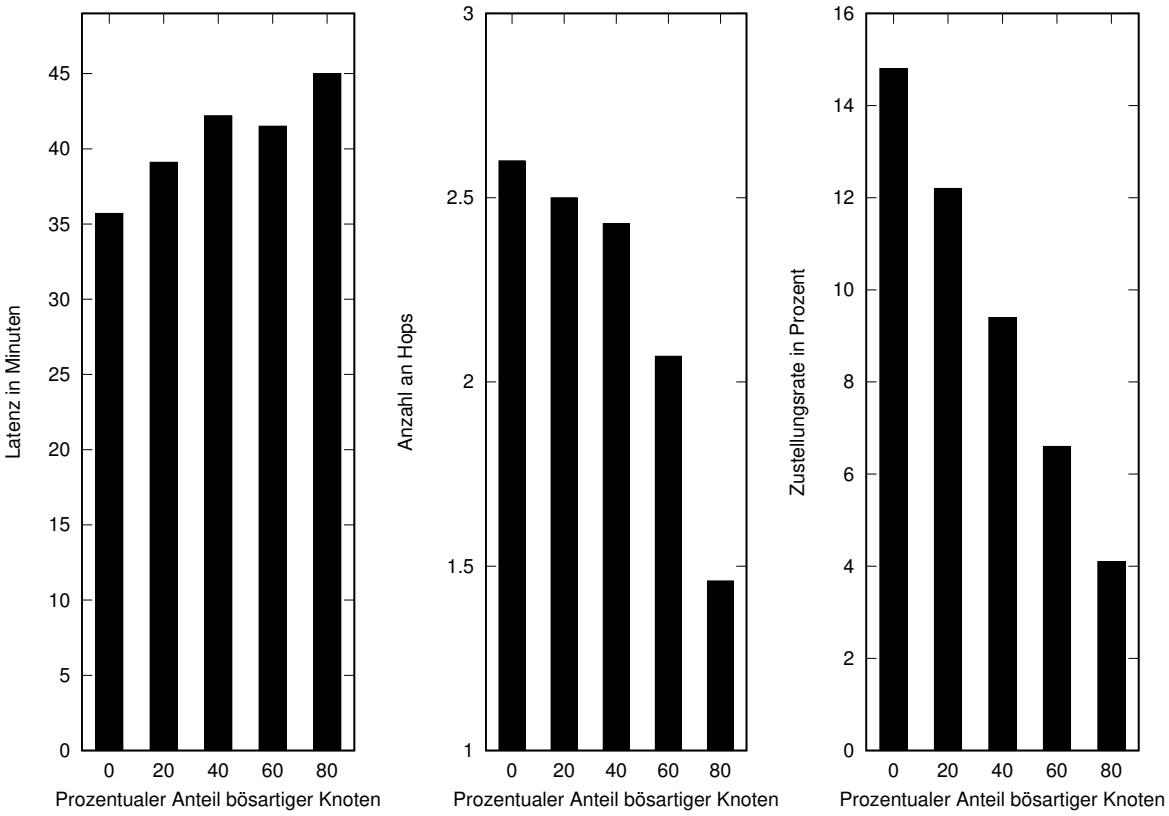
fen von einem beliebigen Knotenpaar betrug in etwa 24 Minuten. Damit ist also der optimale Wert für I_{typ} 24 Minuten. Für das Simulationsszenario bedeutet das auch, dass sich über einen Zeitraum von zwei Stunden alle Knotenpaare, die sich begegnen, dies im Schnitt fünf mal tun. Das ist ausreichend oft, damit die Vorhersagbarkeitstabellen sich mit Werten füllen und PRoPHET gut forwarden kann.
Die gemessenen Metriken ohne bösartige Knoten sind allerdings nicht so gut, wie erwartet, was in Abbildung 5.1 zu erkennen ist. Die Latenz ist sehr hoch und die Zustellungsrate sehr niedrig.
Nach Analysieren der Netzwerksituation ist folgendes aufgefallen. Es seien nun A ein Knoten, der ein Paket aus seinem Puffer an Zielknoten D weiterleiten möchte, und B ein Knoten in der Nachbarschaft von A. Es traten zwei merkwürdige Fälle auf:

1. $P(A,D)$ war fast immer größer als oder gleich $P(B,D)$

2. Beim Routen wurden in kurzen Zeitintervallen häufig mehrfach identische Pakete mit demselben A, B und D betrachtet.

Der erste Fall sorgt dafür, dass Pakete nicht oft an Nachbarknoten weitergeleitet werden und könnte gegebenenfalls dadurch entstehen, dass die Knotenbewegung und Ping-Nachrichten nicht gleichmäßig genug randomisiert sind. Knoten könnten so gegebenfalls häufiger Pakete weiterleiten, für deren Zielknoten sie bereits einen Eintrag in der Vorhersagbarkeitstabelle besitzen.
Der zweite Fall sorgt dafür, dass redundante Informationen verarbeitet werden, da für identische Pakete und identische A, B und D bezüglich des Routings immer dieselbe Entscheidung getroffen wird. Der zweite Fall könnte durch mehrere Gründe auftreten. Es könnten mehrere identische Ping-Nachrichten versendet worden sein oder aber Pakete könnten mehrfach aus dem Puffer gelesen werden. Die genaue Ursache konnte nicht aufklärt werden.
Da es außerhalb des zeitlichen Rahmens dieser Arbeit liegt und nicht Teil des Themas ist, wird diese Problematik nicht behoben, allerdings in den Auswertungen berücksichtigt.
Durch diese zwei Fälle verschlechtern sich die gemessenen Metriken, da sich eine Vielzahl von Paketen im Puffer der Knoten anhäuft, welche nie zum Ziel gebracht werden. Dies ist deutlich zu sehen, da im Simulationsszenario ohne bösartige Knoten es zu Pufferüberläufen kommt, obwohl der Puffer bis zu 1000 Pakete fassen kann. Allerdings ist immer noch der relativen Einfluss der bösartigen Knoten auf die Metriken gut zu sehen, was das Hauptziel dieser Arbeit ist.

Abbildung 5.1: Metriken für unmodifiziertes PRopHET, Angriff 4.1.1a



5.1 TTL Min

In Abbildung 3.1 ist beschrieben, dass ein Paket nur dann gerouted wird, wenn die Anzahl der bereits besuchten Hops geringer ist als die TTL des Pakets, wobei die TTL eines normalen Knotens 20 beträgt. Die bösartigen Knoten setzen jetzt ihre TTL auf 0, nachdem diese Bedingung überprüft wird. Das bedeutet, dass alle Pakete, die an einen bösartigen Knoten gesendet werden, noch einmal zum nächsten Hop gesendet und danach verworfen werden.
Die Metriken, die sich am stärksten verschlechtern, sind die Anzahl der Hops und die Zustellungsrate. Das ist dadurch erklärt, dass ein Paket nur noch dann erfolgreich an seinen Zielknoten geliefert werden kann, wenn der gesamte Übertragungspfad keine bösartigen Knoten

beinhaltet. Die Latenz ist eine interessante Metrik für diesen Angriff, da die bösartigen Knoten gleichzeitig einen positiven und negativen Einfluss auf sie haben. Weil nur noch kurze Pfade erfolgreich zum Ziel führen, sinkt die Latenz. Weil die Anzahl der Pfade die zum Ziel führen sinkt, müssen die Pakete länger in den Puffern der Knoten warten, also steigt die Latenz. Je nachdem wie stark der positive und negative Einfluss auf die Latenz ist, verbessert oder verschlechtert sie sich. Abbildung 5.1 zeigt dies, da die Latenz sich an einer Stelle sogar verbessert, wenn die Anzahl der bösartigen Knoten von 40% auf 60% erhöht wird.

5.2 Falsche Vorhersagbarkeitstabelle + TTL Min

Die bösartigen Knoten tragen jetzt, zusätzlich zum Ändern der TTL, falsche Einträge in ihre Vorhersagbarkeitstabelle ein. Es wird zum Zeitpunkt, wenn die Gleichungen 3.1, 3.2 und 3.3 aus Kapitel 3 verwendet werden, nicht das Ergebnis berechnet, sondern der Wert 0,99999 eintragen.
Die Ergebnisse der Metriken für diesen Angriff sind in Abbildung 5.2 zu sehen. Die Metriken für Angriff 4.1.1c sehen fast genauso aus wie die für Angriff 4.1.1a, da die bösartigen Knoten in beiden Fällen ihre TTL manipulieren. Allerdings hat das Ändern der Vorhersagbarkeitstabelle definitiv auch negative Konsequenzen, was durch Betrachten von Abbildung 5.3 erkenntlich wird. Dort sind die Metriken notiert, welche gemessen werden, wenn die bösartigen Knoten nur noch die Tabelle manipulieren und nicht mehr die TTL.
Das Ändern der Vorhersagbarkeitstabelle bewirkt, dass sehr viele Pakete an die bösartigen Knoten versendet werden, welche diese dann sofort verwerfen. Das hat für dieses Simulationsszenario allerdings auch positive Auswirkungen, da aufgrund zuvor beschriebener Problematiken es häufig zu Pufferüberläufen kommt, welche reduziert werden, wenn viele Pakete, anstatt im Puffer zu verweilen, an die bösartigen Knoten gesendet werden. Wenn also das Manipulieren der TTL mit dem Manipulieren der Vorhersagbarkeitstabelle kombiniert wird, werden so viele Pakete verworfen, dass die Puffer nicht mehr so stark überlaufen. Somit addieren sich die negativen Auswirkungen von diesen beiden Änderungen nicht so stark, wie vielleicht vorher anngenommen werden könnte.
Insgesamt zeigt sich, dass sowohl das Manipulieren der TTL als auch der Vorhersagbarkeitstabelle starke negative Auswirkungen auf die Netzwerkqualität haben. Besonders die Anzahl der benötigten Hops und die Zustellungsrate verschlechtern sich.

Abbildung 5.2: Metriken für unmodifiziertes PRopHET, Angriff 4.1.1c

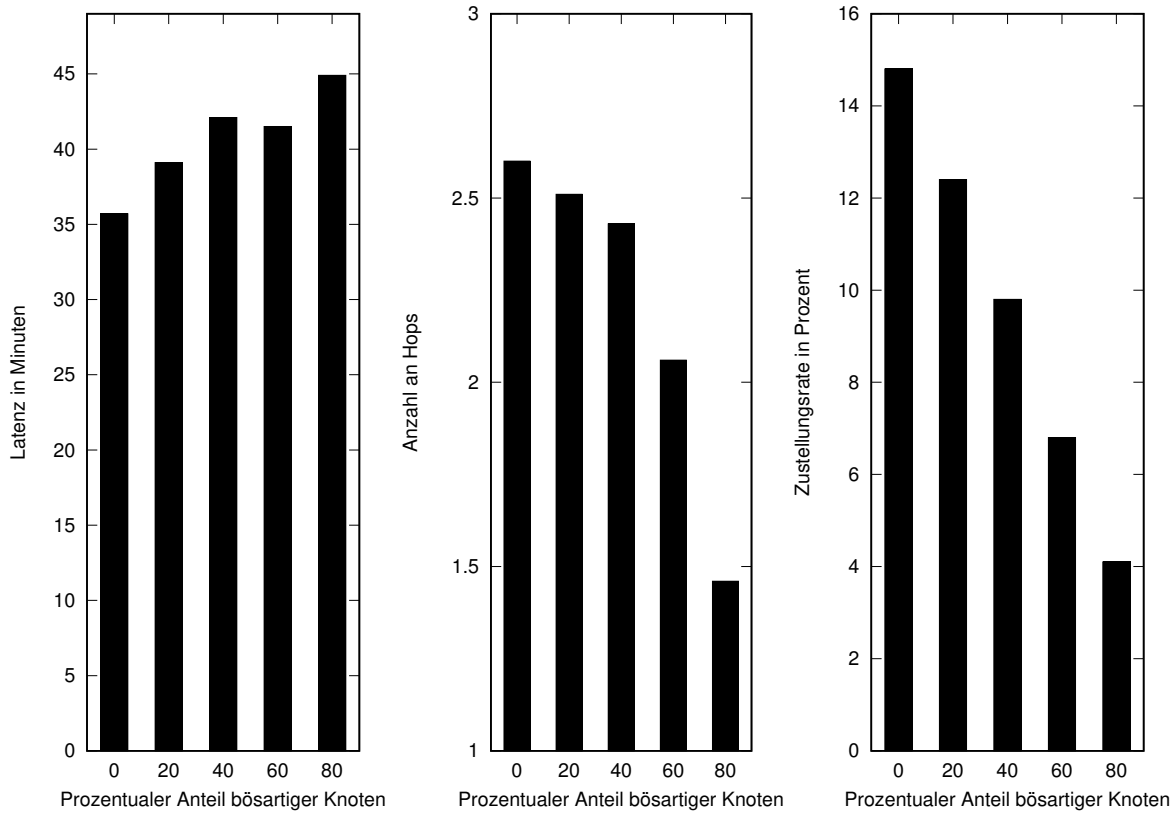
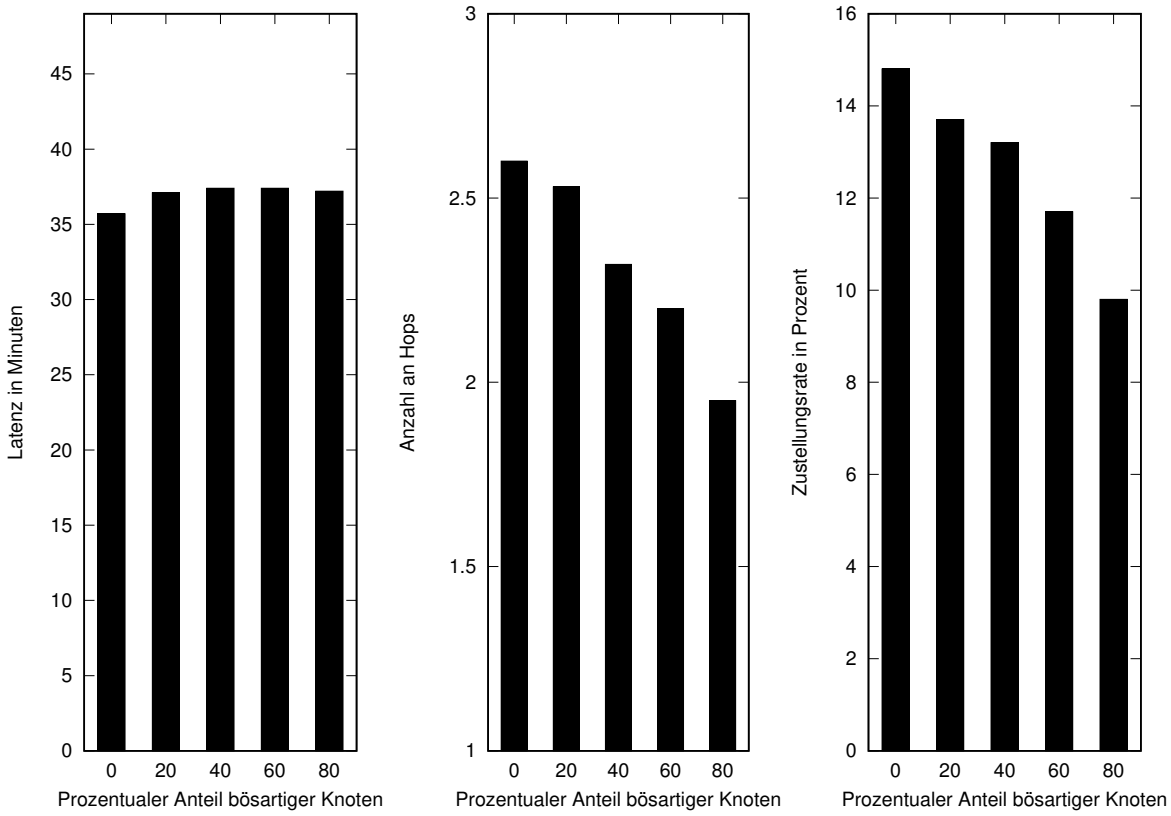


Abbildung 5.3: Metriken für unmodifiziertes PRopHET, falsche Vorhersagbarkeitstabelle



Kapitel 6

Implementierung der PRoPHET Modifikationen

Der generelle Ansatz, der zum Unterbinden der Auswirkungen der bösartigen Knoten verwendet wird, ist der folgende. Es werden Kriterien definiert, die durch normale Knoten kaum zu erfüllen sind. Erfüllt ein Knoten eins dieser Kriterien wird er als bösartig identifiziert. Hat ein normaler Knoten einen bösartigen Knoten identifiziert, speichert er dessen Netzwerk-ID in einer Liste. Danach sollen die negativen Auswirkungen der bösartigen Knoten unterbunden werden, indem die Knoten bei jedem Kontakt mit einem anderen Knoten zuerst überprüfen, ob dieser in der Liste der bösartigen Knoten eingetragen ist. Falls ja, wird der bösartige Knoten für das Routing ignoriert. Dieses gesamte Verfahren wird nur von den normalen Knoten durchgeführt.

6.1 TTL Min

Um bösartige Knoten zu identifizieren wird jedes Mal, wenn ein Paket eintrifft, von den empfangenden Knoten überprüft, ob für das Paket gilt Message.TTL < minValidTTL, wobei minValidTTL auf zwei gesetzt wurde. Es macht wenig Sinn eine TTL zu verwenden, die kleiner als zwei ist, da so die Pakete in fast allen Netzwerken nicht gut ans Ziel gelangen würden. Deswegen kann ein Absender derartiger Pakete als bösartiger Knoten verdächtigt werden. Falls so ein Paket empfangen wird, wird der Absender in die Liste der bösartigen

Knoten eingetragen und die TTL wieder auf den standardmäßigen Wert 20 gesetzt, um zu versuchen das Paket doch noch ans Ziel zu bringen.

6.2 Falsche Vorhersagbarkeitstabelle + TTL Min

Die falschen Einträge in der Vorhersagbarkeitstabelle der bösartigen Knoten haben zwei Effekte. Zum Einen werden mehr Pakete an die bösartigen Knoten geleitet, so dass dadurch die bösartigen Knoten mehr Pakete verwerfen können, zum Anderen verbreiten sich die falschen Einträge auch bei den normalen Knoten, aufgrund der Transitivität.

Es wurde eine Formel entwickelt, mit welcher eine obere Schranke für valide Tabelleneinträge berechnet werden kann, indem bestimmte netzwerkabhängige Attribute miteinander verrechnet werden. In Tabelle 6.1 sind Konstanten angegeben, die in Tabelle 3.1 noch nicht mit Werten versehen waren oder noch nicht existiert haben.

Tabelle 6.1: Konstanten, deren Werte und Bedeutungen

Konstante	Wert	Bedeutung
$ageInterval$	1 Minute	Zeitraum in welchem die Tabelleneinträge einmal altern
M_{typ}	5	Typische Anzahl an Begegnungen zweier Knoten
I_{typ}	24 Minuten	Siehe Kapitel 3

Die Idee ist nun folgende, jeder Tabelleneintrag kann nur durch die drei Gleichungen 3.1 3.2 oder 3.3 modifiziert werden, wobei 3.2 die Werte nur reduzieren kann. Zusätzlich gilt, dass in 3.3 der Wert von $P(A,B) \cdot P(B,D) \cdot \beta$ selten hohe Werte erreicht, da dafür sowohl $P(A,B)$ als auch $P(B,D)$ hohe Werte haben müssten. Außerdem ist durch β der Wert durch 0,9 nach oben hin beschränkt, wohingegen die Gleichung in 3.1 bei mehrfacher Verwendung gegen eins konvergieren kann. Somit wird mit hoher Wahrscheinlichkeit eine präzise obere Schranke für die Tabelleneinträge erzeugt, indem der Tabelleneintrag berechnet wird, der entstehen würde, wenn zwei Knoten sich M_{typ} mal in einem Zeitabstand von I_{typ} begegnen und den Tabelleneintrag mit 3.1 aktualisieren, bevor einer der Knoten das Netzwerk verlässt.

M_{typ} wurde auf fünf gesetzt, da I_{typ} 24 Minuten und die Simulationszeit 120 Minuten ist, also von fünf Begegnungen zweier Knoten ausgegangen werden kann.
Es seien nun die Knoten, die sich begegnen, A und B. Der Tabelleneintrag $P(A,B)$, der bei der ersten Begegnung dieser Knoten entsteht, wird standardmäßig mit 0,5 initialisiert, danach wird der Wert noch vier mal mit 6.1 erhöht und vier mal mit 6.2 reduziert.

$$P(A,B) = P(A,B)_{old} + (1 - \delta - P(A,B)_{old}) \cdot P_{encounter_{max}} \tag{6.1}$$

$$P(A,B) = P(A,B)_{old} \cdot \gamma^{\frac{I_{typ}}{ageInterval}} \tag{6.2}$$

Das Ergebnis für diese obere Schranke war etwa 0,95, ein Wert der von normalen Knoten praktisch nie erreicht wird.
Nachdem also nun eine obere Schranke für valide Tabelleneinträge definiert wurde, untersuchen alle normalen Knoten, ob ein Verbindungspartner einen solchen Eintrag besitzt. Falls ja, wird dieser Eintrag nicht verwendet um die eigene Tabelle zu aktualisieren, um das Verbreiten von den Tabellenwerten der bösartigen Knoten zu unterbinden. Zusätzlich wird ein Knoten, der einen solchen Eintrag propagiert hat, in die Liste der bösartigen Knoten aufgenommen.

Kapitel 7

Simulation mit modifiziertem PRoPHET

Es wird dasselbe Simulationsszenario betrachtet, welches in Kapitel 5 beschrieben wurde, nur wird jetzt die modifizierte Variante von PRoPHET verwendet.

7.1 TTL Min

In Abbildung 7.1 ist erkennbar, wie sich die Metriken nach Implementierung des Lösungsansatzes deutlich verbessern. Die normalen Knoten besitzen nach einer gewissen Zeit eine gut gefüllte Liste von bösartigen Knoten und können diese dann beim Routing umgehen.

7.2 Falsche Vorhersagbarkeitstabelle + TTL Min

Abbildung 7.2 zeigt, wie sich die Metriken verhalten, wenn die bösartigen Knoten nur ihre Tabelle verfälschen, aber nicht die TTL, unter Verwendung des Lösungsansatzes für manipulierte Tabelleneinträge. Die Metriken verbessern sich durch den Lösungsansatz praktisch überhaupt nicht, werden an manchen Stellen sogar schlechter. Wenn die Ursache dafür gefunden werden soll, sollte in Erwägung gezogen werden, dass das alleinige Ändern der Tabelleneinträge durch die bösartigen Knoten bewirkt, dass das Routing sehr randomisiert abläuft. Kein Knoten weiß mehr wirklich, welcher Kontaktpartner denn nun wirklich das Paket gut

ans Ziel bringen kann. Es wird also immer noch uneingeschränkt gerouted, allerdings ist es dann nur noch Zufall, ob ein Paket sein Ziel erreicht. Es könnte in manchen Einzelfällen sogar sein, dass ein falscher Tabelleneintrag dafür sorgt, dass ein Paket besser zum Ziel gelangt. Dass im Durchschnitt das Manipulieren von den Tabelleneinträgen einen negativen Einfluss auf die Metriken hat, ist jedoch in Kapitel 5 in Abbildung 5.3 zu sehen.
Ein potenzieller Grund für die Ergebnisse des Lösungsansatzes könnte sein, dass die normalen Knoten durch das Ignorieren der bösartigen Knoten beim Routing so viele Routingpartner verlieren, dass das alleinige Routen unterhalb der normalen Knoten nicht viel effizienter ist, als das randomisierte Routen durch die bösartigen Knoten.
In Abbildung 7.3 sind die Ergebnisse für die Metriken zu sehen, die gemessen werden, wenn die bösartigen Knoten sowohl die TTL als auch die Tabelleneinträge manipulieren und die normalen Knoten beide Lösungsansätze verwenden. Es fällt auf, dass sich zwar Latenz und Anzahl der Hops nicht stark verbessern, allerdings die Zustellungsrate stark verbessert werden konnte.

Abbildung 7.1: Metriken für modifiziertes PRopHET, Angriff 4.1.1a

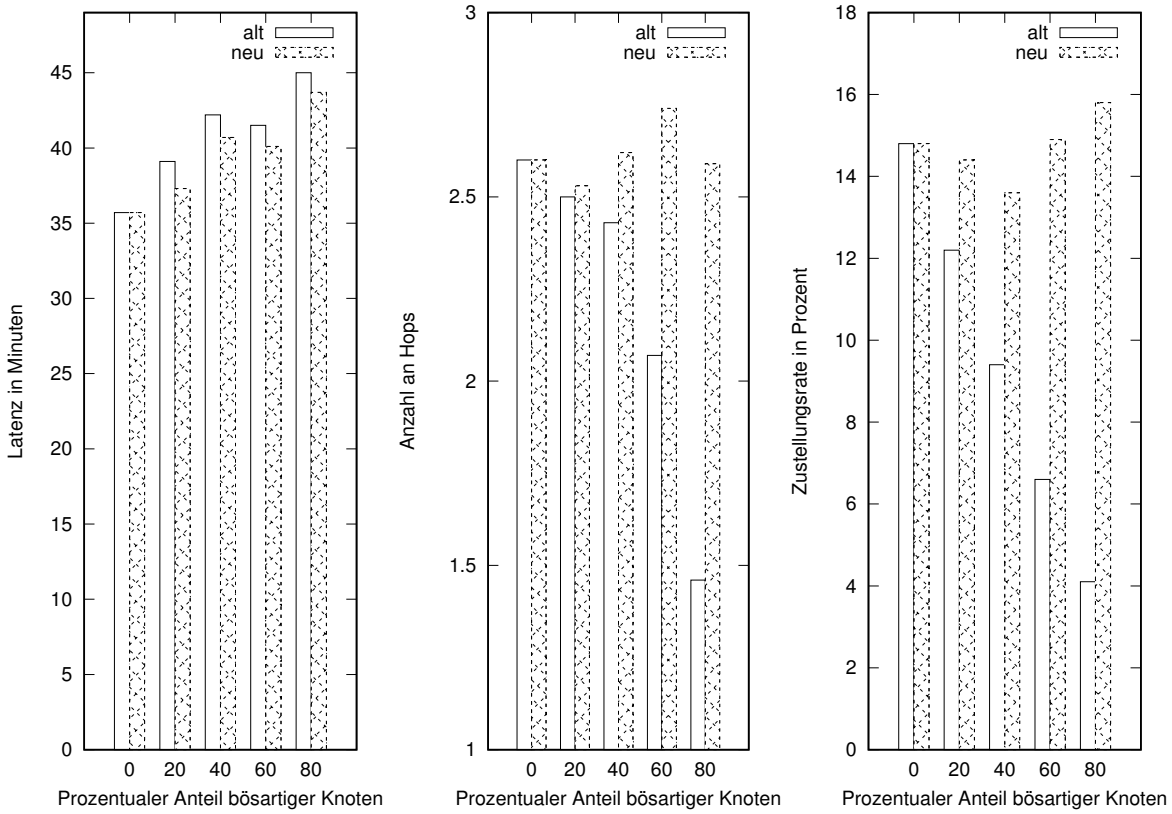


Abbildung 7.2: Metriken für modifiziertes PRopHET, falsche Vorhersagbarkeitstabelle

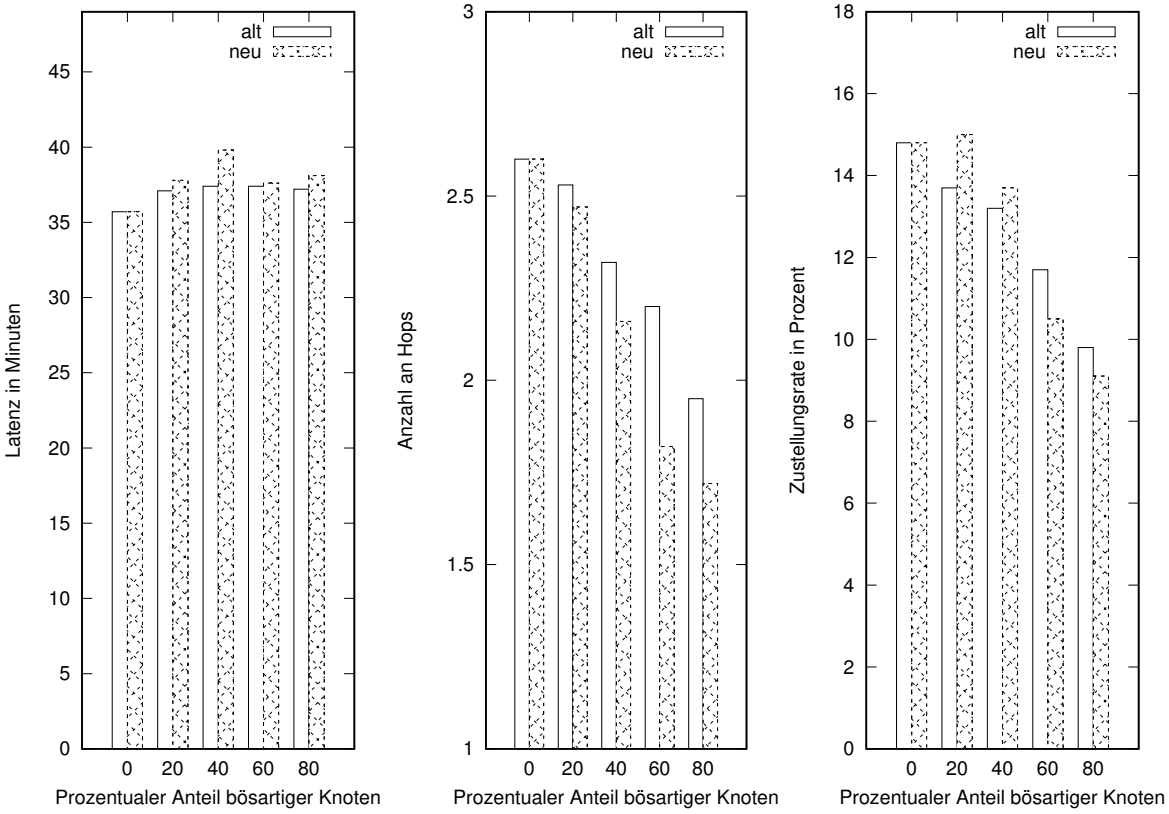
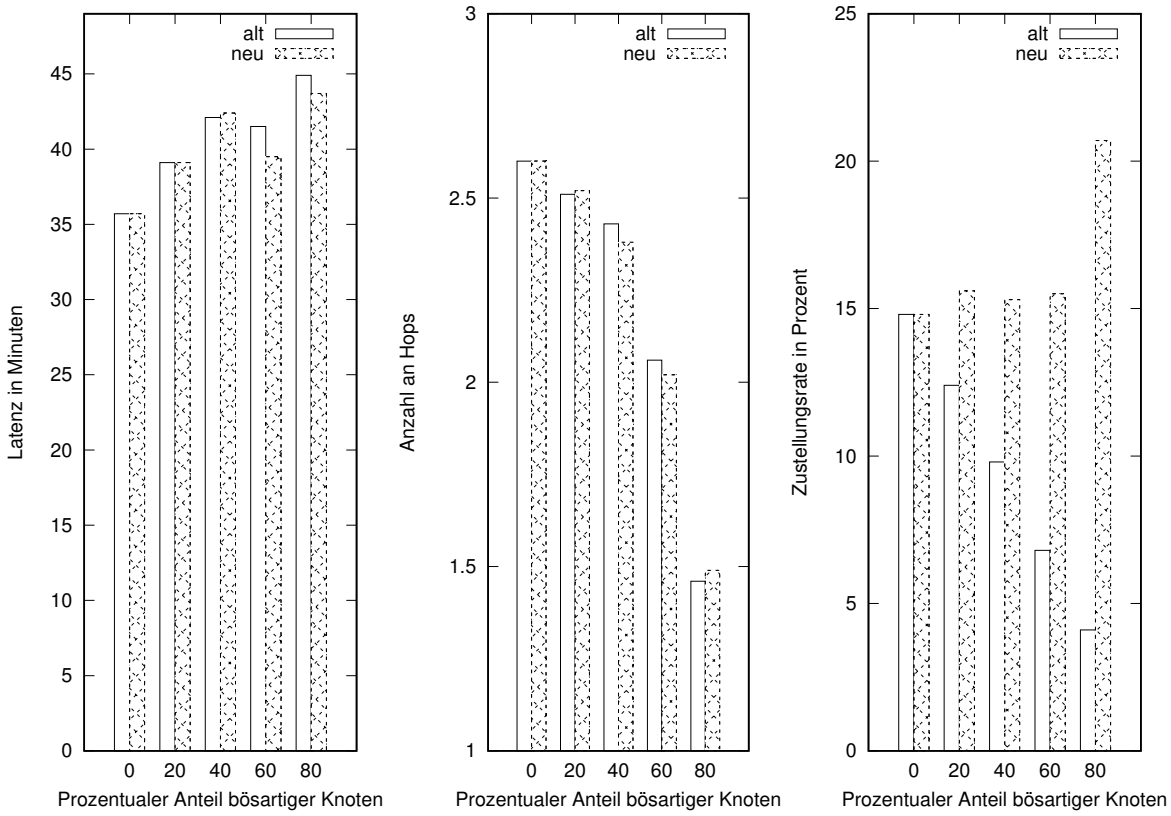


Abbildung 7.3: Metriken für modifiziertes PRopHET, Angriff 4.1.1c



Kapitel 8

Zusammenfassung

Das Thema dieser Arbeit war es, das Routing-Protokoll PRoPHET gegenüber bösartigen Knoten in opportunistischen Netzwerken sicherer zu machen.
Die bösartigen Knoten, die in dieser Arbeit betrachtet wurden, haben die TTL von Paketen und die Vorhersagbarkeitstabelleneinträge manipuliert, so dass sich die Netzwerkqualität verschlechtert hat.
Das Entwickeln von Gegenmaßnahmen bezüglich Angriffen in Netzwerken kann grob in zwei Schritte unterteilt werden.

1. Identifizieren der Angreifer

2. Einschränken des Einflusses der Angreifer

Das Identifizieren der Angreifer wurde realisiert, indem die Validität der propagierten Vorhersagbarkeitstabellenwerte mittels einer berechneten oberen Schranke überprüft wurde. Bei Überschreiten dieser Schranke wurde der propagierende Knoten als bösartig deklariert. Dasselbe wurde auch für die TTL der ankommenden Pakete gemacht, wobei für diese keine Schranke berechnet wurde, sondern alle TTLs unter zwei als bösartig deklariert wurden.

Das Einschränken des Einflusses der bösartigen Knoten wurde realisiert, indem alle als bösartig deklarierten Knoten beim Routen und Aktualisieren der Vorhersagbarkeitstabellen ignoriert wurden.

8.1 Fazit

Die implementierten Gegenmaßnahmen haben teilweise gute Ergebnisse geliefert, besonders das Modifizieren der TTL konnte gut unterbunden werden, allerdings haben sich die Metriken bezüglich des Angriffs, welcher die Vorhersagbarkeitstabelle manipuliert, durch die vorgestellte Gegenmaßnahme nicht stark verbessert. Die Ergebnisse sind allerdings nicht nur abhängig von der Gegenmaßnahme, sondern auch vom Simulationsszenario. Es könnte sein, dass die Resultate sich durch das Verändern des Simulationsszenarios verbessern.

8.2 Ausblick auf die Zukunft

Die Vorgehensweise in dieser Arbeit, um PRoPHET zu verbessern, ist nur eine von vielen Möglichkeiten die Problematik zu behandeln. Es könnte an einigen Stellen anders vorgegangen oder die verwendeten Ansätze erweitert werden.

8.2.1 PeerfactSim.KOM

In dieser Arbeit wurde der Simulator PeerfactSim.KOM (siehe [Gra11]) verwendet, welcher als open project von vielen Studenten und Angestellten der Heinrich-Heine-Universität und der TU Darmstadt erweitert wird. Der Simulator könnte an vielen Stellen noch verbessert werden, vor allem für den Bereich opportunistische Netzwerke, da der Simulator ursprünglich nur für P2P-Netzwerke entwickelt wurde. Der Simulator hatte zur Zeit des Schreibens dieser Arbeit nur zwei Versionen von PRoPHET implementiert gehabt, einmal so wie in dieser Arbeit präsentiert und einmal eine lokationsbasierte Version. Es könnten auch spezielle Arten von Datenverkehr für opportunistische Netzwerke implementiert werden, die sich für konkrete realistische Szenarien eignen. In dieser Arbeit wurden randomisierte Ping-Nachrichten betrachtet, was ein eher unrealistisches, jedoch einfach zu analysierendes, Szenario ist.

8.2.2 Angriffe

In dieser Arbeit wurden von den sieben vorgestellten Angriffen zwei behandelt. Es könnte jetzt noch betrachtet werden, wie die verbleibenden fünf Angriffe behandelt werden oder ob andere Lösungsansätze für die bereits behandelten verwendet werden. Besonders das Behandeln des Wormhole Angriffs bietet sich als interessantes Thema an, da in [RCYC10] bereits gute Ansätze präsentiert wurden. Die Problemstellung besteht also nur noch darin, die Lösung zu vervollständigen und in PeerfactSim.KOM zu implementieren.

8.2.3 Simulationsszenario

In dieser Arbeit wurde ein möglichst realistisches Simulationsszenario gewählt und versucht die von PRoPHET verwendeten Konstanten so optimal wie möglich zu wählen, um als Ausgangspunkt für die Metrik-Analyse besonders gute Startwerte zu haben. Allerdings gibt es noch viel mehr Szenarien die definiert werden könnten. Es könnten die Simulationsfläche, die Anzahl der Knoten oder die Art von Daten, die die Knoten gegenseitig an sich schicken, verändert werden und noch vieles mehr. Es sollte allerdings beim Definieren eines Simulationsszenarios beachtet werden, dass große Simulationsszenarien viel Rechenkapazität benötigen. Die hier vorgestellten Simulationen wurden auf einem Laptop, welcher über einen Intel Core I7-2670QM verfügt, berechnet. Die Simulationsdauer war im Schnitt 10-15 Minuten, wobei immer fünf Simulationen parallel liefen. Falls ein Szenario betrachtet wird, welches nur etwas größer ist als das in dieser Arbeit betrachtete, dann erhöht sich die Simulationszeit bereits drastisch.

Literaturverzeichnis

[BG16] BIALON, Raphael; GRAFFI, Kalman: Misrouted Prophecy–On the Impact of Security Attacks on PRoPHET. In: *European Conference on Parallel Processing* Springer, 2016, S. 296–308.

[CGMP10] CONTI, Marco; GIORDANO, Silvia; MAY, Martin; PASSARELLA, Andrea: From opportunistic networks to opportunistic computing. In: *IEEE Communications Magazine* 48 (2010), Nr. 9.

[DL09] DISTL, Bernhard; LEGENDRE, Franck: *On Opportunistic Networking Security*. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.561.958&rep=rep1&type=pdf`. Version: 2009. [In keinem Journal veröffentlicht, nur online; Stand Februar 2018]

[Gra11] GRAFFI, Kalman: PeerfactSim. KOM: A P2P system simulator—Experiences and lessons learned. In: *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on* IEEE, 2011, S. 154–155.

[LDDG12] LINDGREN, A.; DORIA, A.; DAVIES, E.; GRASIC, S.: *Probabilistic Routing Protocol for Intermittently Connected Networks*. RFC 6693 (Informational). `http://www.ietf.org/rfc/rfc6693.txt`. Version: 2012 (Request for Comments). [Online; Stand 6. Februar 2018]

[NAP14] NAGRATH, Preeti; ANEJA, Sandhya; PUROHIT, GN: Flooding Attack in Delay Tolerant Network. In: *International Journal of Emerging Technology and Advanced Engineering* (2014), July.

[RCYC10] REN, Y.; CHUAH, M. C.; YANG, J.; CHEN, Y.: Detecting wormhole attacks in delay-tolerant networks [Security and Privacy in Emerging Wireless Networks]. In: *IEEE Wireless Communications* 17 (2010), October, Nr. 5, S. 36–42. http://dx.doi.org/10.1109/MWC.2010.5601956. DOI 10.1109/MWC.2010.5601956. ISSN 1536–1284.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 6. März 2018 Andreas Lisowsky

Diese CD enthält:

- eine *pdf*-Version der vorliegenden Bachelorarbeit
- die LaTeX- und Grafik-Quelldateien der vorliegenden Bachelorarbeit samt aller verwendeten Skripte
- die Quelldateien des verwendeten Simulators PeerfactSim.KOM
- die ausgelesenen Messdaten der Simulationsergebnisse
- die Websites der verwendeten Internetquellen