Simulatoren für drahtlose Netzwerke

Mateusz Khalil
Betreuer: Thomas Hillebrandt

Freie Universität Berlin, 15.Januar 2010

Simulation spielt eine wichtige Rolle, um Systeme und deren Verhalten zu analysieren. In dieser Arbeit wird zunächst eine Einführung in die Netzwerksimulation gegeben, wobei vertieft auf den Effekt von Abstraktion eingegangen wird. Es werden Kriterien für die Simulation drahtloser Netzwerke beschrieben und mit denen verdrahteter Netzwerke verglichen. Anschließend werden die Simulatoren ns2, JiST/SWANS und ShoX analysiert und anhand der Kriterien ausgewertet. Zuletzt wird bewertet, ob sich die Werkzeuge für die Analyse und Entwicklung von Netzwerkprotokollen in ausgewählten Netzwerken eignen.

1 Einführung in die Netzwerksimulation

Da der Bedarf an drahtloser Kommunikation steigt, ergibt sich ein neues Spektrum an Anwendungen für mobile Endgeräte. Mit dem Auftreten neuer Systeme und Kommunikationsformen, werden neue Protokolle und Algorithmen entwickelt. Gegenstand der Forschung sind verteilte, selbstorganisierte und dynamische Systeme. Das Verhalten dieser Systeme lässt sich intiuitiv oftmals nicht einschätzen oder bewerten, sodass auf andere Formen der Analyse zurückgegriffen werden muss.

1.1 Analyseformen

Traditionsgemäß existieren drei Analyseformen von Netzwerksystemen: Testbeds, Simulation und mathematische Analyse, von denen Simulation die verbreiteste ist [CASE09]. Bei Testbeds werden Protokolle auf Basis echter Hardware in einer experimentellen Umgebung getestet. Sie liefern realistische Ergebnisse, da sie den physikalischen Gesetzmäßigkeiten unterliegen und somit keine Abstraktion einfließt. Die Aufwandskosten sind enorm, da Hardware benötigt wird. Oftmals werden in der Forschung nicht existierende Geräte betrachtet, sodass ein Testbed nicht in Frage kommt. Die Wiederholung eines Experiments ist sehr aufwändig, da die Hardware eventuell neu kalibriert bzw. auf den Ausgangszustand zurückgesetzt werden muss. Da man auf eine bestimmte Umgebung

eingeschränkt ist, sind die Ergebnisse nur für die vorliegenden Gegebenheiten und nicht allgemein gültig. Die mathematische Analyse weist die höchste Abstraktion vor. Die Schwierigkeit bei diesem Ansatz ist die Komplexität der Layer und Protokolle. Oftmals müssen die Systeme vereinfacht werden, was zu falschen Ergebnissen führen kann.
Bei der Simulation wird ein Modell anhand eines realen Systems erstellt. Anschließend wird versucht das Verhalten des realen Systems anhand von Rückschlüssen bezüglich des Modells zu ermitteln. Der entscheidende Vorteil von Simulation ist das Verhältnis zwischen Qualität der Ergebnisse und der Komplexität bzw. Kosten. Rahmenbedingungen können mit wenig Aufwand geschaffen werden. Die Wiederholung des Experimentes ist einfach.
Prinzipiell existieren zwei unterschiedliche dynamische Simulationsarten[1]. Einerseits gibt es Continuous Dynamic Simulation, welche mithilfe von Differentialgleichungen beschrieben wird. Das System wird kontinuierlich in der Zeit neu berechnet. Die zweite Simulationsart beschreiben Discrete Event Simulators, wobei der Simulationsfortschritt durch Auftreten von Ereignissen entsteht. Discrete Event Simulation eignet sich hervorrangend für die Simulation von Netzwerken, wobei beispielsweise das Schlüsselereignis *Paket wird gesendet* zu einem Simulationsfortschritt führt. Dies erklärt auch, weshalb die verbreitesten Netzwerksimulatoren diese Simulationsart bevorzugen.

1.2 Effekte von Abstraktion

Abstraktion hat unterschiedliche Auswirkungen auf die Qualität der Simulationsergebnisse. Vor der Simulation wird ein Modell bezüglich eines realen Systems erstellt. Um möglichst reale Ergebnisse zu erzielen, ist es wichtig, das Modell mit möglichst vielen Details auszustatten. Eine vollkommen realistische Simulation ist nicht möglich, da sonst die Realität samt Photonen, Elektronen etc. simuliert werden müsste. Je detailierte die Modelle sind, umso komplexer ist die Implementierung und langsamer die Ausführung. Die Schwierigkeit der Simulationsdesigner ist somit die Auswahl der korrekten Abstraktiontiefe bzw. die Entscheidung, wie detailiert die Modelle sein müssen. Schließlich weiß dieser oft nicht, welche Größen einen Einfluss auf die Simulationsergebnisse haben können. Deshalb versucht man oft ein Modell zu validieren, indem man es mit Werten aus realen Umgebungen oder detaillierten Simulationen vergleicht. Während des Validierungsprozesses wird klar, welche Einflussgrößen dem Modell notwendigerweise hinzugefügt werden müssen.
Eine hohe Abstraktionsebene eignet sich hervorragend für die Beseitigung unwesentlicher Nebeneffekte. So kann beispielsweise die Anwendung unterschiedlicher Algorithmen besser auf eine bestimmte Eigenschaft hin untersucht werden.
[DETAIL]vergleicht anhand eines Beispiels energiesparende ad-hoc Routingalgorithmen. Dabei wird ein stark abstrahiertes Energiemodell benutzt, welches davon ausgeht, dass Knoten im Leerlauf keine Energie verbrauchen. In einer weiteren Simulation wird das Energiemodell erweitert. Es zeigt sich, dass das vereinfachte Modell im Gegensatz zum erweiterten Energiemodell falsche Vorhersagen trifft und folglich inkorrekt ist. Als wei-

[1] Ausgenommen sind Spezial- und Hybridlösungen

teres Beispiel beschreibt [DETAIL] die örtliche Lokalisierung eines Knotens mithilfe von Beacons (Knoten mit integriertem GPS). Bei dieser Betrachtung ist das Wellenausbreitungsmodell wichtig, da der Knoten mithilfe der Signalstärke bezüglich der Beacons seine Position bestimmen soll. Es wird ein sehr einfaches mathematisches Modell verwendet und mit einem realen Experiment verglichen. Erstaunlicherweise liegt die Abweichung der Vorhersagen im Vergleich zur Realität durchschnittlich bei ca. 15%. Dieser Fall zeigt, dass es auch Anwendungen gibt, die unempflindlich bzgl. des Abstraktionsgrades sind. Zuletzt weist [DETAIL] auf robuste Algorithmen hin, welche ebenfalls unempfindlich im Hinblick auf den Abstraktionsgrad der jeweiligen Anwendungsschicht sind. Dies ist auf die Definition von Robustheit zurückzuführen, welche besagt, dass der Algorithmus trotz unerfüllter Bedingungen gut funktioniert.

1.3 Anforderung an Netzwerksimulatoren

Für einen Protokolldesigner ist es aufgrund der Anzahl existierender Simulatoren schwierig, einen für seinen Bedarf günstigen Simulator auszuwählen. Deshalb werden hier Kriterien beschrieben, welche bei der Auswahl helfen sollen.
Zunächst ist der Abstraktionslevel der vom Simulator angebotenen **Modelle** wichtig. Wie im Abschnitt *Effekte von Abstraktion* beschrieben, ist das Detail für die Qualität der Ergebnisse ausschlaggebend. In dieses Kriterium fließt ebenfalls die Validierung der jeweiligen Modelle ein. Ist ein Modell validiert, erhöht es die Qualität der Ergebnisse.
Eine weitere wichtige Eigenschaft ist die **Performance**. Dabei versteht man unter Performance nicht nur die benötigte Zeit, um eine Simulation durchzuführen, sondern auch die maximale mögliche Anzahl an eingefügten Netzwerkknoten (Speicherverbrauch). Grundsätzlich kann man sagen, dass sie von den Faktoren Abstraktionsgrad der Modelle, Architektur der Simulationssoftware und von Parallelisierbarkeit abhängig ist. Wie in [PERF03] beschrieben, sind sequentielle Simulatoren, welche die Zeit und Knoten sequentiell betrachten, den parallelisierbaren Simulatoren hinsichtlich der Performance stark unterlegen. In [PERF03] und im IEEE 1516 wird gezeigt, wie Parallelisierung durch Kopplung mehrerer unterschiedlicher Simulatoren für extrem große Experimente erreicht werden kann. Da dies einen Mehraufwand darstellt und die Gesamtkomplexität erhöht, soll dennoch die Parallelität auf Ebene eines Simulators betrachtet werden.
Bereits unterstütze Protokolle, Erweiterbarkeit und Programmierung, **Tools, Dokumentation und Support** unterstützen den Protokolldesigner bei der Entwicklung der Simulation und sie erleichtern ihm die Bedienung der Simulationssoftware. Als weiteres Kriterium kommt **Visualisierung und Auswertung** hinzu, welches dem Anwender bei der Interpretation der Simulationsergebnisse und der Beseitigung von Protokollfehlern helfen soll. Oftmals wird verlangt, dass ein Simulator mit einem vorhandenen Netzwerk kommunizieren soll, sodass neben der virtuellen nun auch reale Transmission existiert. Diese Eigenschaft fällt unter den Begriff der Emulation.

1.4 Besonderheiten der Simulation drahtloser Netzwerke

Zunächst können die Kriterien für die Simulation von verdrahteten Netzwerken übernommen werden. Im Vergleich zu verdrahteten Netzwerken bestehen Unterschiede hinsichtlich der Protokolle und Modelle. Die Natur drahtloser Netzwerke äußert sich in fluktuierenden Verbindungen und zumeinst batteriebetriebenen mobilen Endgeräten. Da auch ein anderes Übertragungsmedium gegeben ist, unterscheiden sich die Protokolle auf der physikalischen als auch auf der Sicherungsschicht voneinander. Es existieren zusätzliche Modelle, die den Energieverbrauch, die Mobilität und Wellenausbreitung umfassen. Im jüngeren Feld der Netzwerkesimualtion, der Simulation drahtloser Netzwerke, besteht das Problem der schlechten Qualität der Simulationsergebnisse [EXP09]. Die Validierung erfolgt bei Simulation verdrahteter Netzwerke stets indem Ergebnisse des jeweiligen Modells mit experimentellen Werten aus der Realität verglichen werden. Im Gegensatz dazu lassen sich Experimente in drahtlosen Netzwerken bei gleichen Verhältnissen nicht einfach reproduzieren, wie im einfachen Aufbau in [EXP09] beschrieben und deutlich wird. Problematisch ist außerdem, dass die Modelle zumeist nicht für alle Umgebungen anwendbar sind. Als Beispiel diene das Wellenausbreitungsmodell, es existieren Modelle für geschlossene Gebäude, offene Terrains, Städte und viele weitere. Es müssen viele Effekte berücksichtigt werden wie Interferenz, Reflexion und Shadowing, was die Komplexität der Wellenausbreitungsmodelle erhöht.

2 Analyse ausgewählter Simulatoren

Analysiert und anschließend in bestimmten Anforderungsumgebungen verglichen werden die Discrete Event Simulatoren ns-2, JiST/SWANS und ShoX. Obwohl ShoX kein bekannter Simulator ist, wird dieser ebenfalls in die Analyse einbezogen. ShoX ist im Gegensatz zu den anderen ausgewählten Simulatoren von Grund auf für die Simulation drahtloser Netzwerke entwickelt worden ist.

2.1 ns2

ns2 ist der bekannteste und verbreiteste Simulator, der unter der GNU-Lizenz steht. Er besitzt eine objektorientierte Architektur und basiert auf C++. Mit der Skriptsprache OTcl kann das Experiment konfiguiert und verwaltet werden. Möchte man ein neues Protokoll integrieren, so muss im Kern C++ Code hinzugefügt und OTcl-Konfigurationsdateien angepasst werden. Der Vorgänger von ns2, ns, wurde 1989 als Simulator für verdrahtete Netzwerke entwickelt. In ns2 wurde eine Bibliothek zur Verfügung gestellt, die die Simulation drahtloser Netzwerke ermöglicht.

2.1.1 Modelle und Protokolle für drahtlose Netzwerke

Die Mobilität und Erstellung einer zwei dimensionalen physikalischen Umgebung (mit Hindernissen) wird gewährleistet. Es können Position, Bewegungsrichtung und Geschwin-

digkeit je Knoten angepasst werden. Pfade zufällig zu generieren ist ebenfalls möglich. Die Kofiguration der Knoten bezüglich der Mobilät muss vor dem Experiment erstellt werden. Desweiteren ist eine Emulation durchführbar. Es existiert ebenfalls ein Energie- und Wellenausbreitungsmodell. Letzteres umfasst folgende Modelle:

- Free Space Model
- Two-Ray Ground Reflection Model
- Shadowing Model

Auf physikalischer Ebene sind unterschiedliche Fading Models implementiert. Außerdem bietet ns2 ein Data Diffusion Model, welches häufig in Simulationen von Sensornetzwerke benutzt wird. ns2 stellt desweiteren auf MAC- und physikalischer Ebene Modelle zu 802.11, 802.16, impulse-radio ultra-wide band (IR-UWB) model, 3G, Cognitive Radio Network model, TDMA und viele weitere bereit. Für weitere Informationen siehe [CCODE], wo auch die große Anzahl an vorhandenen Routingalgorithmen aufgelistet ist. Wichtig zu erwähnen ist, dass das 802.11 Modell nicht validiert ist.

2.1.2 Performance

Die Architektur von ns2 ist weder in der Simulationszeit noch bezüglich von Knotenverbänden parallelisierbar. In [CASE09] wird gezeigt, dass ns2 nicht skaliert. PDNS, was für Parallel/Distributed Network Simulator steht, wurde vom George Institute of Technology entwickelt. Es ist eine Erweiterung von ns2, welche die Performanceprobleme durch den Aspekt der Rechenleistung kompensieren soll. Es verwendet eine konservative Synchronisationsmethode, sodass viel Nachrichten-Overhead im Netzwerk vorhanden ist. Dadurch muss kein Rollback-Mechanismus implementiert werden und die Änderungen an ns2 sind somit minimal. Viele schnelle parallelisierbare Simulatoren verwenden hingegen optimistische Synchronisationsstrategien. Tritt bei dieser im Verlauf der Simulation ein Konflikt auf, wird ein entsprechendes Rollback durchgeführt, um diesen zu lösen. In [PERF03] wird an einem Experiment gezeigt, dass PDNS annähernd skaliert [Abbildung 1]. Als Beispiel für die Leistungsfähigkeit wird ein Experiment beschrieben. Dabei generieren 1.0 Million Webbrowser einen HTTP-Traffic. Insgesamt existieren 1.1 Millionen Knoten. Um 300 Sekunden des realen Experiments nachzubilden werden 541 Sekunden benötigt. Anhand dieser Daten sieht man, dass PDNS unter Umständen echtzeitfähig ist.

2.1.3 Erweiterbarkeit und Programmierung

Aufgrund des objektorientieren Ansatzes lässt sich ns2 schnell erweitern. Dennoch geben einige Autoren [COMPA] an, dass die Einarbeitungszeit vergleichsweise hoch ist und viele Konzepte und die kaum verbreitete Skriptsprache OTcl erlernt werden müssen, um ns2 entsprechend zu konfiguieren und erweitern.

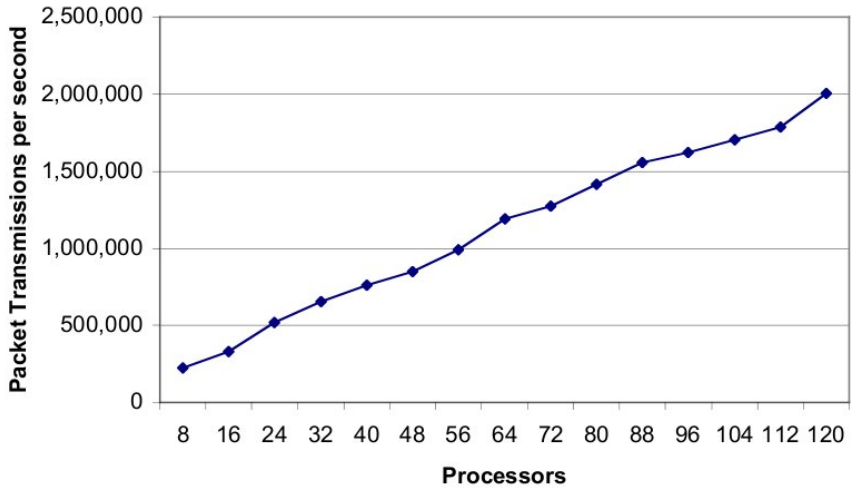


Abbildung 1: Skalierbarkeit von PDNS

2.1.4 Tools, Dokumentation und Support

Die Dokumentationen in ns2 sind ausführlich und berücksichtigten detailierte Aspekte, die vom QuickGuide bis zur Architekturbeschreibung von ns2 reichen. Die Community ist sehr aktiv, ebenso in der Mailinglist als auch im öffentlich zugänglichen Wiki. ns2 wird ständig weiterentwickelt; das letzte Release is ns2.34 und wurde am 17.Juli 2009 veröffentlicht. Problematisch ist dagegen die begrenzte Anzahl von Tools in Bezug auf die Konfiguration des Experiments. Diese müssen zumeist vom Benutzer selbst entwickelt werden.

2.1.5 Visualisierung und Auswertung

Network animator (nam) heißt das einzige Visualisierungswerkzeug, welches für ns2 zur Auswahl steht und in Tcl geschrieben wurde. Anhand der Log-Dateien, die während der Simulation eines Experiments erstellt werden, visualisiert ns2 Knoten, Links, Bewegungen, Packete, etc. aber auch Zustandsänderungen von Knoten. Für die Auswertung und das Plotten der Statistik kann das *ns2graph: RPI NS-2 Graphing and Statistics Package* benutzt werden.

2.2 JiST/SWANS

Java in Simulation Time (JiST) ist ein Open-Source Dicrete Event Simulator. Scalable Wireless Ad Hoc Network Simulator (SWANS) ist eine Bibliothek, die vollständig auf JiST aufsetzt und hauptsächlich für die Simulation von MANETs (Mobile Ad Hoc

Networks) entwickelt worden ist. Sie bietet zusätzliche Modelle und Protokolle für die Simulation drahtloser Netzwerke an. Darüberhinaus existiert das Projekt SWANS++, welches JiST/SWANS um VANETs (Vehicular Ad Hoc Networks) erweitert. Die Lizenz besagt, dass für akademische Zwecke der Gebrauch der Simulation frei zugänglich ist.

2.2.1 Modelle und Protokolle für drahtlose Netzwerke

SWANS bietet unterschiedliche Wellenausbreitungs- und Mobilitätsmodelle. Bezüglich der Wellenausbreitung existieren folgende Modelle:

- Free Space Model
- Two-Ray Ground Pathloss Model

Auf phyisikalischer Ebene können Rayleigh oder Rician Fading benutzt werden. Die Mobilitätsmodelle können in Form von zufälliger Bewegung, zufällige Wegpunkte und Teleporting ausgewählt werden. Dabei kann die Bewegung zur Simulationslaufzeit verändert werden. In JiST/SWANS kann man für einen Knoten die Intervalle einstellen, in denen die Position neu berechnet werden soll. Somit entfernt man sich etwas vom Ansatz der Discrete Event zur Time Based Simulation. Man behält alle Strukturen bei. Der Scheduler wird mit wiederkehrenden Position-Update-Events gefüllt. Die Entwickler von JiST/SWANS nahmen diese Performanceeinbuße hin (Positionen werden aktualisiert, obwohl evtl. kein Bedarf an der Information besteht), um Flexibilität zur Simulationslaufzeit zu ermöglichen. Es stehen viele Routingalgorithmen zur Disposition, die in [JISTB] eingesehen werden können. Als MAC-Protkoll steht lediglich 802.11b zur Verfügung. Wichtig zu erwähnen ist die fehlende Validierung jeglicher in SWANS implementierter Modelle, welche die Qualität der Ergebnisse zunächst in Frage stellt. So werden in [JVSN2] zwei Routingalgorithmen in ns2 als auch in JiST/SWANS anhand des gleichen Experimentaufbaus simuliert. Die Implementierungen des Routingalgorithmus sind in beiden Simulatoren gleich. Bereits bei diesem einfachen Versuchsaufbau entstehen Unterschiede von bis zu 30% [Abbildung 2]. Dabei bewerten die Autoren, dass JiST/SWANS ähnliche Ergebnisse zu ns2 produzieren kann, wobei die Unterschiede durch die unterschiedlichen Implementierungen der unteren Layer zu erklären sind.

2.2.2 Performance

Es wurden bisher kaum wissenschaftliche Beiträge zur Performance von JiST/SWANS veröffentlicht. In [JISTB] wird eine Simulation des NDP-Protkolls durchgeführt und mit ns2 und GloMoSim verglichen. Dabei wird einerseits deutlich, wie stark sich zwei unterschiedlichen Wellenausbreitungsmodellen auf die Performance auswirken können [Abbildung 3]. Andererseits wird überraschenderweise der große Performancevorteil von JiST/SWANS gegenüber ns2 und GloMoSim gezeigt. Nicht nur in Bezug auf den benötigten Speicher, sondern auch im Durchsatz ist JiST/SWANS den Konkurrenten überlegen. Eine Analyse der Qualität der Ergebnisse liegt dagegen nicht vor. Es ist zunächst verwunderlich, dass ein auf Java basierender Simulator C (GloMoSim Parsec-Simulator)

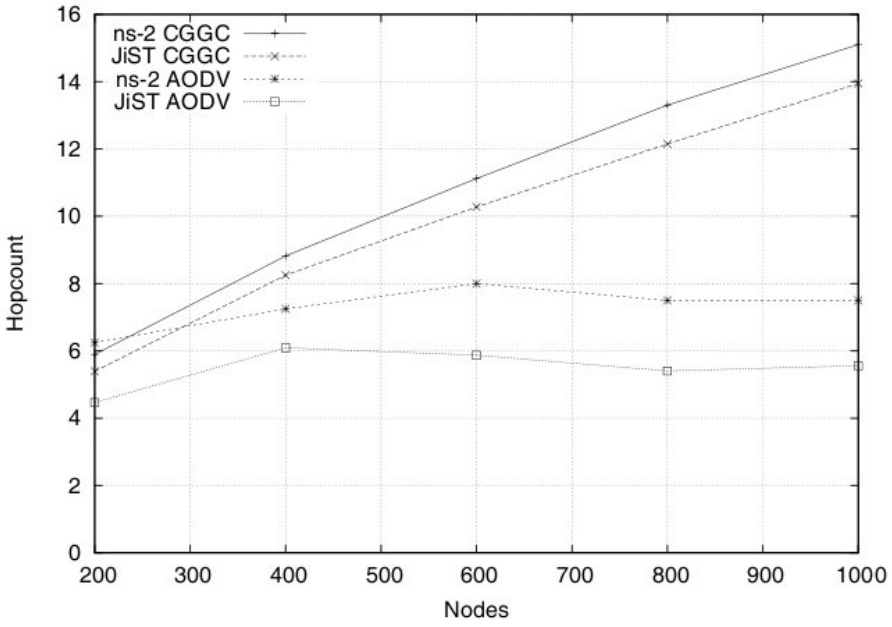
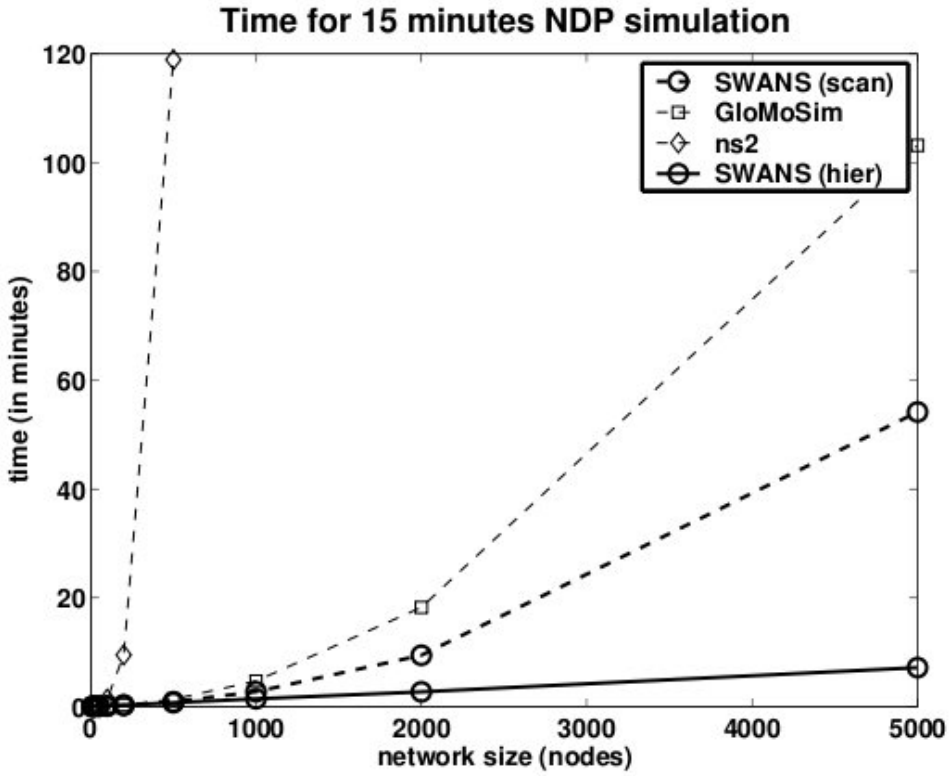


Abbildung 2: Unterschiede der Ergebnisse zwischen JiST/SWANS und ns2 bei gleicher Implementierung



(b) linear scale

Abbildung 3: Performancevergleich zwischen ns2, JiST/SWANS, GloMoSim. Der Performancevorteil von SWANS soll gezeigt werden.

als auch C++ (ns2) so deutlich überlegen ist, da Java prinzipiell einen Performancenachteil gegenüber den genannten Programmiersprachen besitzt. Die Entwickler von JiST/SWANS erheben den Anspruch, dass der Simulator skaliert (Scalable Wireless Ad Hoc Network Simulation). Da die Events als Threads ausgeführt werden, ist eine Parallelisierung auf Ebene der Events aufgrund der Unterstützung durch die JVM möglich. Es besteht auch die Möglichkeit der verteilten Simulation.

2.2.3 Architektur

Die Architektur sieht vor, dass die Simulation mit einem gängigen Java Compiler kompiliert wird. Daraufhin wird der Byte Code modifiziert, um anschließend durch den Simulationskernel ausgeführt zu werden [Abbildung 4]. Die Modifizierung des Byte Codes

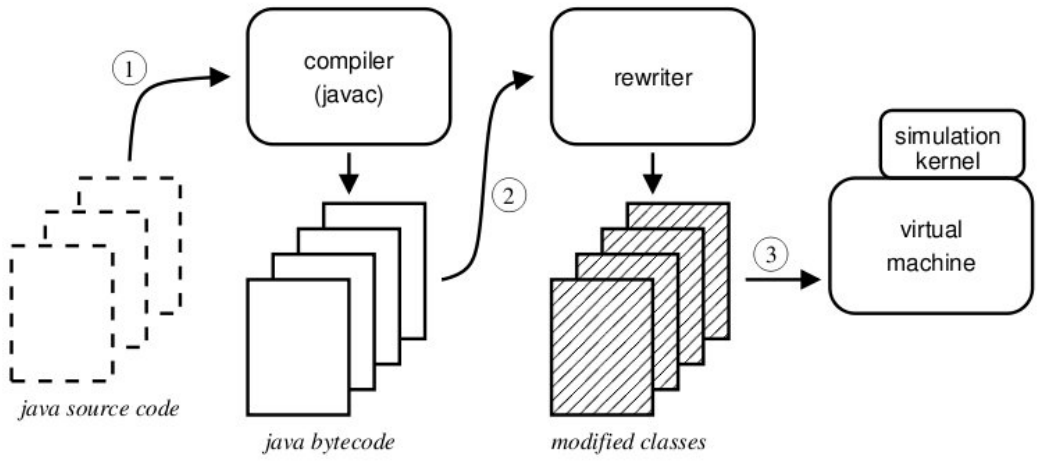


Abbildung 4: Funktionsweise von JiST

führt dazu, dass sich die Objekte an die Simulationszeit, statt der realen Zeit, orientieren. Überlicherweise orientiert sich die JVM an der der realen Zeit; dabei sind Programmfortschritt und reale Zeit voneinander unabhängig. Man kann also ein Programm zweifach gleichzeitig ausführen und zu einem bestimmten Zeitpunkt (reale Zeit) können beide Programme (durch den Einfluss des Schedulers, Interrupts, etc.) einen unterschiedlichen Programmfortschritt aufweisen. Andere Discrete Event Simulatoren benötigen ebenfalls eine Event-Warteschlange, die nach der Simulationszeit sortiert ist, um das korrekte nächste Event bezüglich der simulierten Zeit auswählt. Der Unterschied in JiST ist, dass die Zeitsemantik der Java Standard Language hinzugefügt und eine darunterliegende VM verwendet wird. Ein Fortschritt der Simulationszeit für ein Event

entsteht nur, wenn das Programm auf eine *sleep(time)* Funktion trifft. Somit benötigt jeder andere Befehl bezüglich der Simulationszeit keine Zeit. Für die anderen Events ensteht hinsichtlich der Simulationszeit kein Seiteneffekt. Der Fortschritt der globalen Simulationszeit und die Verwaltung inklusive Schedulings der Event-Warteschlange erfolgt durch den JiST-Kernel.

2.2.4 Erweiterbarkeit und Programmierung

JiST basiert vollständig auf Java; für Konfigurationen wird keine zusätzliche Skriptsprache verwendet, obwohl sich Jython anbietet. Wegen der Benutzung von Java, bietet JiST/SWANS ebenfalls dessen Vorteile; Plattformunabhängigkeit, Typsicherheit, Objektorientierung und Parallelisierbarkeit. Die Programmierung und Erweiterbarkeit ist wegen der Objektorientierung und der klar definierten JiST-API schnell erlern- und anwendbar.

2.2.5 Tools, Dokumentation und Support

Die Dokumentation von JiST/SWANS basiert auf der Arbeit von [JISTB]. Desweiteren gibt es keine Alternativen vergleichbaren Umfangs. Die Toolverfügbarkeit beschränkt sich auf wenige Tools. Im Bereich VANETs hingegen gibt es DUCKS und STRAW. DUCKS ist ein Framework zum automatischen Simulieren und Evaluieren der Ergebnisse. Mit STRAW kann ein Straßenmobilitätsmodell erstellt werden. Der Kern des Simulators wird nicht mehr weiterentwickelt. Eine Mailinglist, Wiki oder Forum existiert nicht.

2.2.6 Visualisierung und Auswertung

Der *JiST Graphical User Interface Event Viewer* dient zur Visualisierung der generierten Log-Dateien der Simulation. Das Programm visualisiert die Events in Tabellen und nicht in einem Koordinatensystem. Daher kann man mit dem Event Viewer die Entwicklung und den Fortschritt der Events verfolgen; die Knoten und Transaktionen werden aber nicht visualisiert. Es ist somit kein Visualisierungstool im klassischen Sinne. Ein Auswertungstool ist nicht vorhanden.

2.3 ShoX

A Scalable Ad Hoc Simulator (ShoX) ist ein in Java geschriebener quelloffener Discrete Event Simulator. Er wurde explizit als Simulator drahtloser Netzwerke entwickelt und steht unter der GNU-Linzenz zur Verfügung. In der Architektur unterscheidet er sich von den anderen Simulatoren durch den zusätzlichen Layer *AirModule*, welcher mögliche Interferenzen ermittelt und Zustände der Knoten festhält (wie senden und empfangen). Mithilfe dieser Informationen bestimmt das AirModule anhand eines InterferenceHandler, ob ein Paket, Frame, etc. tatsächlich den Empfänger erreicht. Das Ziel der Entwick-

ler von ShoX ist es, einen Simulator zu entwerfen, mit dem mit möglichst wenig Aufwand eine Simulation drahtloser Netzwerke konfiguriert und durchgeführt werden kann.

2.3.1 Modelle und Protokolle für drahtlose Netzwerke

Es existieren alle grundlegenden Wellenausbreitungs-, Mobilitäts- und Energiemodelle, die für die Simulation drahtloser Netzwerke benötigt werden. Auf der physikalischen und MAC-Ebene steht nur der Standard 802.11b zur Disposition. Desweiteren gibt es einige Routingalgorithmen, darunter auch geographische. Die Schwäche von ShoX ist die kleine Anzahl an verfügbaren Modellen bezüglich des OSI-Referenzmodells.

2.3.2 Performance

Ähnlich zum Konzept von JiST/SWANS ist ShoX aufgrund von der JVM parallelisierbar. Es existiert auch ein Layer, der für das verteilte Simulieren zuständig ist. Obwohl ShoX den Anspruch auf Skalierbarkeit erhebt, gibt es bislang keine Veröffentlichung die diesen Aspekts bestätigt oder untersucht hat.

2.3.3 Erweiterbarkeit und Programmierung

Die objektorientierte Ausrichtung der Architektur ermöglicht ein einfaches Erweitern der Simulationssoftware. Alle Schichten des OSI-Referenzmodells liegen als abstrakte Klassen vor. Um ein neues Protokoll zu entwerfen, genügt schon das Implementieren einer dieser Klassen. XML-Dateien werden für die Konfiguration der Simulation eingesetzt. ShoX besitzt ein sehr mächtiges Werkzeug zur Erstellung von Szenarios, betrachte dazu den Abschnitt *Tools, Dokumentation und Support*. In [COMPA] wurde SPAN in unterschiedlichen Simulatoren implementiert, darunter ns2 und ShoX. Die Einarbeitungszeit in ShoX war die geringste und es wurden im Vergleich zu ns2 25% weniger Code benötigt. Obwohl keine ausführliche Dokumentation existiert, genügen die API-Beschreibungen, um eine effiziente Übersicht über ShoX zu erhalten. Java bietet außerdem den Vorteil, dass der Speicher mithilfe der Garbage Collection automatisch verwaltet wird, was den Aufwand des Entwickler minimiert.

2.3.4 Tools, Dokumentation und Support

Obwohl die Dokumentation spärlich ist, existieren im Wiki Beispiele für die Implementierung neuer Protokolle, Erstellung von Statistiken und Konfiguration von Simulationen. Die schlechte Dokumentation stellt, wie im Abschnitt *Erweiterbarkeit und Programmierung* erwähnt, kein Hindernis für das effiziente Benutzen von ShoX dar. Dennoch wurde im Oktober 2009 ein Wiki eingerichtet, welches das Dokumentationsdefizit kompensieren soll. Es besteht die Möglichkeit direkt mit den Entwicklern per Mail in Kontakt zu treten. ShoX bietet ein sehr mächtiges Werkzeug zu Konfiguration der Simulation. Implementiert man ein eigenes Protokoll, welches bei der Initialisierung von Parametern

abhängig ist, so kann man eine Annotation bezüglich der benötigten Variable hinzufügen. Das Configuration Panel erkennt die Annoation und interagiert bei der Simulationskonfiguration mit dem Benutzer über die GUI. Als weiteres Feature ist die Generierung der Landkarte zu nennen. Diese lässt sich nämlich mithilfe einer Vektorgrafik (SVG) generieren; dabei werden die Hindernisse erkannt und erstellt.

2.3.5 Visualisierung und Auswertung

Das Visualisierungs- und Auswertungswerkzeug ist sehr mächtig. Wie in anderen Simulatoren auch, werden zunächst die Log-Dateien der Simulation benötigt, um eine Visualisierung zu erstellen. Die betreffenden Events, welche dargestellt werden sollen, können bereits im Configuration Panel vor der Simulation ausgewählt werden. [Abbildung 5] zeigt das Visualisierungswerkzeug. Es exisitieren drei Bereiche. Die untere Leiste dient dazu, um zu verschiedenen Simulationszeitpunken zu springen, die Wiedergabegeschwindigkeit zu regeln oder in der Landkarte zu zoomen. Im REC_ON Modus kann man das Darstellungsfenster aufzeichnen und im MPEG-Format speichern. Im zentralen Bereich werden Knoten, Links und Pakete visualisiert. Desweiteren werden im linken Bereich Debug-Informationen in Textform, die zu der jeweiligen Simulationszeit ausgegeben wurden, aufgelistet. Das in ShoX integrierte Statistiktool JFreeChart ist sehr nützlich. Es enthält viele unterschiedliche Darstellungstypen und Funktionen. Beispielsweise können statistische Daten bezüglich eines oder mehrerer Knoten betrachtet werden.

3 Bewertung

Die drei vorgestellten Simulatoren besitzen unterschiedliche Stärken und Schwächen, die im Anhang in der Tabelle *Stärken und Schwächen der vorgestellten Simulatoren* zusammengefasst werden. In vielen Arbeiten wird auf die Performanceschwäche von ns2 hingewiesen. Aufgrund von PDNS lässt sich auch ns2 parallelisieren, sodass dieser Nachteil ausgeglichen werden kann. Die Betrachtung, inwiefern sich die Simulationswerkzeuge für die Entwicklung von Netzwerkprotokollen in drahtlosen Netzwerken eignen, muss differenziert durchgeführt werden. Schließlich eignen sich alle drei Simulatoren für bestimmte Szenarien besser als die Konkurrenten. Entscheidend ist, ob weitere Netzwerkprotkolle vorausgesetzt werden.
In der Forschung werden oft nicht existierende Protokolle entwickelt. Deshalb ist es für den Forschenden wichtig, möglichst einfach Modifikationen und Erweiterungen durchzuführen. Er benötigt Tools, die ihn bei der Auswertung der Simulationsergebnisse unterstützen. Dabei ist die Qualität der Ergebnisse kein zwingendes Kriterium, da oftmals nur Hinweise auf ein bestimmtes Verhalten gesucht werden. In diesem Szenario ist ShoX die eindeutig beste Wahl. Schließlich besitzt es mächtige statistische und visuelle Werkzeuge. Aufgrund der Architektur lassen sich Protokolle mit geringem Aufwand integrieren. Ist die Qualität und Verfügbarkeit von Modellen von Interesse, so ist ns2 die richtige Wahl. Schließlich bietet ns2 die größte Anzahl an Modellen und Protokollen. Ist das Inte-

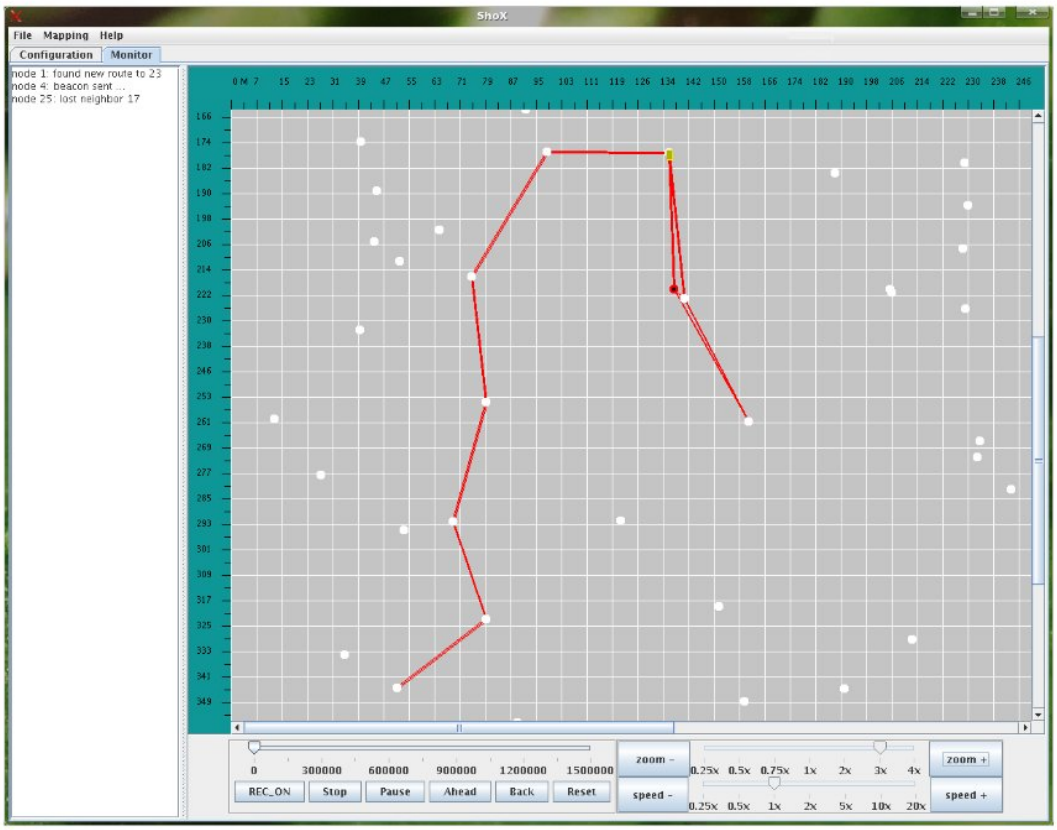


Abbildung 5: Visualisierungswerkzeug von ShoX

resse auf VANETs beschränkt, so ist JiST/SWANS vorzuziehen, da es in diesem Bereich spezialisierte Bibliotheken und Tools anbietet.
Angeführt soll ein weiteres Szenario, bei dem der Bedarf an einer großen Anzahl an zu simulierenden Knoten besteht, etwa wie bei Sensornetzwerken. In diesem Fall gibt es unterschiedliche Kriterien, um einen Simulator auszuwählen. Da diese Netzwerke oftmals nicht mobil sind, empfiehlt es sich ns2 mithilfe von PDNS zu benutzen. Schließlich bietet ns2 viele Routingalgorithmen und Protkolle, die nützlich sein können. Außerdem besteht die Möglichkeit der Emulation, sodass es mit echten Sensornetzwerken interagieren kann. Spielt Mobilität eine entscheidende Rolle, so sollte man JiST/SWANS den Vorzug geben, aufgrund seines flexiblen Mobilitätsmodells und der Performance.
Abschließend wird bewertet, wie sich die Qualität der Simulationsergebnisse im Vergleich zu realen drahtlosen Netzen verhält. Da ns2 die qualitativ detailliertesten Modelle bietet, wird es repräsentativ für die Qualität der Simulationsergebnisse benutzt. Im Experiment in [ACCN2] wurde ein drahtloses, statisches Multi-Hop Ad-hoc-Netzwerk innerhalb eines Gebäudes aufgebaut. Die Ergebnisse vom realen Netzwerk, von der Emulation und Simulation werden verglichen. Das Versenden eines MPEG4-Videos von einem Client zu einem Server innerhalb des Netzwerks wird analysiert. Desweiteren wird ein eigenes Routingprotokoll benutzt. Die Topologien in der Simulation und Emulation sind aufgrund der Benutzung von Shadowing, welches laut [ACCN2] realistische Ergebnisse liefert, gleich. Sie unterscheiden sich bezüglich der realen Topologie um 10%. Das Verhältnis der angekommenen Paket weicht um 0.3% vom realen Wert ab. Zuletzt betrachtet [ACCN2] die Latenz, in welcher bei der Simulation ein Fehler von 57% entsteht. Im Vergleich dazu weicht die Emulation höchstens um 20% ab. Dieses Experiment zeigt bereits, dass die Qualität der Simulationsergebnisse bezüglich der betrachteten Parameter in drahtlosen Netzwerken unterschiedlich ist. Es muss aber das grundsätzliche Problem der Qualität der Eingangsparameter beachtet werden. In [EXP09] wird gezeigt, dass die Eingangsgrößen nicht reproduzierbar sind und fluktuieren. Somit besitzen Simulationen drahtloser Netzwerke bereits bei ihrer Erstellung einen Fehler, der sich auf das Simulationsergebnis unterschiedlich stark auswirken kann. Außerdem wird das Experiment von [ACCN2] innerhalb eines Gebäudes durchgeführt, in welchem viele physikalische Effekte auftreten, die im Shadowing Model nicht betrachtet werden.

4 Zusammenfassung

Simulation bietet einen guten Kompromiss zwischen der Qualität der Ergebnisse und der Kosten. Dabei sind Testbeds für die Analyse drahtloser Netzwerke oftmals ungeeignet, da Experimente in diesen oftmals nicht reproduzierbar sind. Mathematische Modelle bieten häufig einen zu hohen Abstraktionsgrad, sodass Effekte nicht ermittelt werden können. Es wurde beschrieben, wie sich der Abstraktionsgrad der Modelle auf die Qualität der Simulationsergebnisse und auch auf die Performance auswirken kann. Es gibt Fälle, in denen einfache Modelle genügen, um realistische Ergebnisse zu erhalten. Andererseits können Effekte bei Modellen mit zu hohem Abstraktionsgrad nicht auftreten. Je detailierter das Modell, umso mehr Rechenleistung wird benötigt. Anschließend wur-

den Anforderungen an Netzwerksimulatoren entwickelt und um den Aspekt der Drahtlosigkeit erweitert; Modelle und Protokolle, Performance, Erweiterbarkeit und Programmierung, Tools, Dokumentation und Support und Visualisierung und Auswertung. Anhand dieser Kriterien wurden die Vor- und Nachteile der ausgewählten Simulatoren ns2, JiST/SWANS und ShoX ermittelt. Dabei überzeugt ns2 besonders bei den angebotenen Modellen und deren Qualität. Seine Schwäche ist die lange Einarbeitungszeit, die unter anderem mit dem Erlernen von OTcl zusammenhängt. JiST/SWANS weist eine sehr gute Performance vor, besitzt dagegen kein Tool zur Visualisierung und Auswertung. ShoX ist sehr benutzerfreundlich und stellt mächtige Tools zur Verfügung. Es fehlen aber Modelle und deren Validierungen. Zuletzt wurden Szenarien beschrieben in denen gezeigt wurde, dass sich jeder der ausgewählten Simulatoren in bestimmten Fällen besser eignet als die Konkurrenten.
Zuletzt wurde die Qualität der Simulationsergebnisse mit realen drahtlosen Netzwerken verglichen. Dabei stellte sich heraus, dass es ein grundsätzliches Problem bei der Qualität der Eingangsgrößen für die Simulation gibt. Dieser Effekt wird durch fluktuierende Parameter wie Interferenzen und instabile Signalstärken verursacht. Somit können betrachtete Parameter unterschiedliche Qualitäten aufweisen, die im repräsentativen Experiment [ACCN2] zwischen 0.3% und 57% von den realen Werten abweichen.

5 Anhang

ns2	JiST/SWANS	ShoX
+Modelle und Protokolle	+Performance	+Verfügbarkeit von Tools
+Qualität der Ergebnisse	+Architektur	+Architektur
+Dokumentation und Community	+VANETs	+Statistik
+Entwicklung und Erweiterung von ns2		+Visualisierung
+Emulation		+Kurze Einarbeitungszeit
-Verfügbarkeit von Tools	-Verfügbarkeit von Tools	-Protokolle und Modelle
-statisches 2D-Mobilitätsmodel	-Validierung der Modelle	-Validierung der Performance fehlt
-Hohe Einarbeitungszeit	-Weiterentwicklung der Simulationssoftware	

Stärken und Schwächen der vorgestellten Simulatoren

Literatur

[CASE09] S. Mehta and Niamat Ullah and Md. Humaun Kabir and Mst. Najnin Sultana and Kyung Sup Kwak, “A Case Study of Networks Simulation Tools for Wireless Networks“ in Asia International Conference on Modelling Simulation, 2009

[DETAIL] "Effects of Detail in Wireless Network Simulation", John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun-chan Lan, Ya Xu, Wei Ye, Deborah Estrin, Ramesh Govindan. Submitted to SCS Communication Networks and Distributed Systems Modeling and Simulation Conference. September, 2000.

[PERF03] Richard M. Fujimoto, Kalyan Perumalla, Alfred Park, Hao Wu, Mostafa H. Ammar, George F. Riley, "Large-Scale Network Simulation: How Big? How Fast?," Modeling, Analysis, and Simulation of Computer Systems, International Symposium on, p. 116, 11th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'03), 2003

[EXP09] Vincent Lenders, Margaret Martonosi, "Repeatable and Realistic Experimentation in Mobile Wireless Networks," IEEE Transactions on Mobile Computing, pp. 1718-1728, December, 2009

[CCODE] List of contributed code in ns2, http://nsnam.isi.edu/nsnam/index.php/Contributed_Code

[COMPA] Johannes Lessmann, Peter Janacik, Lazar Lachev, Dalimir Orfanus, "Comparative Study of Wireless Network Simulators", International Conference on Networking, pp. 517-523, Seventh International Conference on Networking (icn 2008), 2008

[JISTB] Rimon Barr, "An Efficient, Unifying Approach to Simulation Using Virtual Machines", A Dissertation Presented to the Faculty of the Graduate School of Cornell University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy by May 2004, 2004

[JVSN2] Elmar Schoch, Michael Feiri, Frank Kargl, Michael Weber, "Simulation of Ad Hoc Networks: ns-2 compared to JiST/SWANS", Ulm University, Institute of Media Informatics, 2008

[ACCN2] Svilen Ivanov, Andre Herms, Georg Lukas, "Experimental Validation of the ns-2 Wireless Model using Simulation, Emulation, and Real Network", Institute for Distributed Systems, University of Magdeburg