IPv6 im Netzsimulator NS-3:
Eine Bestandsaufnahme

WS 2011-2012

Christian Schneider
Hochschule Bonn-Rhein-Sieg, Fachbereich Informatik
(christian.schneider1@smail.inf.h-bonn-rhein-sieg.de)

Abstract

In dieser Arbeit wird der Umfang der IPv6 Unterstützung von NS-3 ermittelt. Dazu wird NS-3 auf Implementierungen hinsichtlich IPv6 untersucht. Zur Bewertung der IPv6-Fähigkeit von NS-3 wird diese mit dem Umfang der IPv4 Implementierung in NS-3 verglichen.

1. Problemstellung

Die Simulation von Netzen mit geeigneten Tools zwecks Analyse bringt einige Vorteile mit sich. So können Systeme untersucht werden, die noch nicht vorhanden, zu teuer oder aufwendig sind. Weiterhin können Eingabeparameter jederzeit ausgetauscht werden, was bei der analytischen Untersuchung meist nicht möglich ist. Zudem haben Veränderungen an einem solchen Modell keinen Einfluss auf das bestehende Netz. Bei der Simulation werden Experimente an einem Modell durchgeführt, um Erkenntnisse über das reale Netz/System zu gewinnen.

Bietet ein Netzsimulator (z. B. NS-3) diese Vorteile auch bezüglich IPv6, existieren also bereits Implementierungen von IPv6 in dem Netzsimulator? Ziel der Arbeit ist es zu überprüfen, in welchem Ausmaß IPv6 in dem Netzsimulator NS-3 vorhanden ist und wie umfangreich diese im Verhältnis zu IPv4 ist.

2. NS-3

NS-3 ist ein diskreter ereignisorientierter Netzsimulator. Der Quellcode steht unter der GNU GPLv2 Lizenz, wodurch NS-3 frei verfügbar ist. Hauptanwendungsgebiete sind die Forschung und die Entwicklung. Alle drei Monate wird eine neue (stable) Version von NS-3 veröffentlicht. Diese Arbeit bezieht sich auf die Version 3.12, die im August 2011 veröffentlicht wurde.

NS-3 ist eine neue Version des Simulators NS-2. Im Gegensatz zu NS-2 ist NS-3 rein in C++ geschrieben. Weitere Unterschiede zwischen NS-2 und NS-3, sowie Kritikpunkte an NS-2 werden im Kapitel 2.2. beschrieben. Die erste Version von NS-3 erschien im Juni 2008.

2.1. Community

Bereits im Jahre 2006 wurde die Entwicklung von NS-3 durch die 'National Science Foundation', das 'Institut national de recherche en informatique et en automatique' (INRIA) und das 'Georgia Institute of Technology' gesponsert. Zusätzliche Mittel werden von weiteren Institutionen wie z. B. der 'University of Washington' und der 'University of Porto' zur Verfügung gestellt. An NS-3 arbeiten laut eigenen Angaben bis zu 1000 Entwickler und über 20 'Maintainer', die für einzelne Module verantwortlich sind und weiterführende Aufgaben verfolgen, wie z. B. Code-Überwachung, rechtzeitiges Code-Review, Bugfixing etc. [5].

2.2. Vergleich zu NS-2

Über die Jahre sind an NS-2 verschiedene Kritikpunkte entstanden. Diese beziehen sich hauptsächlich auf Design und Performance. So kann in NS-2 pro Node nur eine IP-Adresse vergeben werden. Außerdem macht es die Kombination von C++ und oTCL schwierig neue Modelle zu implementieren und zu debuggen. Zudem haben sich ungewartete bzw. inkompatible Modelle angesammelt und die Dokumentation wurde vernachlässigt [2].

Diese Punkte haben zur Motivation bzgl. der Neu/Weiterentwicklung von NS geführt. Für die

Interoperabilität hat dies folgende Konsequenzen: NS-2 Skripts funktionieren nicht in NS-3, da NS-2 OTcl als Skripting-Umgebung benutzt, NS-3 hingegen C++ Programme oder Python-Skripts. Models, die in NS-2 existieren (OTcl-Models), sind und werden nicht portiert, da dies einem kompletten Neuschreiben gleichkommen würde [7].

3. Funktionsweise und Architektur

3.1. Hauptkomponenten

NS-3 verfügt über mehrere Hauptkomponenten die eine Kommunikation ermöglichen. Diese sind in Abbildung 1 dargestellt und werden nachfolgend beschrieben.

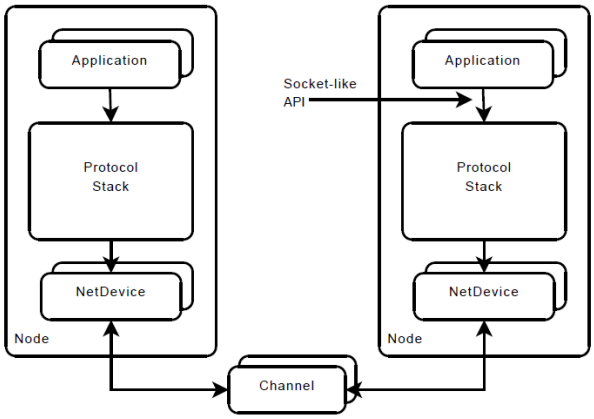

Abbildung 1: NS-3 Hauptkomponenten [2]

Node

Ein Node steht für eine Netzkomponente. Dies kann beispielsweise ein PC, Router, Server etc. sein. Ein Node stellt bildlich somit eine "Box" dar, in der weitere Komponenten, wie z. B. Interfaces (NetDevice) und Applikationen (Application), hinzugefügt werden können. Mehrere Interfaces und Applikationen in einem Node hinzuzufügen, ist hierbei ohne Probleme möglich.

NetDevice/Channel

Ein Interface wird durch eine NetDevice Klasse repräsentiert, das Medium zur Kommunikation (Kabel, Luft) durch einen Channel. Beide Klassen sind abstrakt und besitzen weitere Unterklassen, die entsprechende Technologien darstellen: Zum Beispiel CsmaNetDevice und CsmaChannel für Ethernet Verbindungen, PointToPoint-NetDevice und PointToPointChannel für Punkt-zu-Punkt Verbindungen, sowie WifiNetDevice und WifiChannel für Funkverbindungen. Die NetDevices und die entsprechenden Channels müssen immer zusammen verwendet werden und können nicht vertauscht werden (es kann beispielsweise kein WifiNetDevice mit einem PointToPointChannel verbunden werden) [4].

Application

Zur Darstellung von Anwendungen, die Informationen (Pakete) mit anderen Anwendungen auf entfernten Nodes austauschen, verwendet NS-3 die Klasse Application. Diese Paket-Generatoren können einem Node hinzugefügt werden und kommunizieren dann mit einer Menge von Stacks.

Helper/Container

Um die Arbeit zu erleichtern existieren sogenannte Helper/Container APIs. Diese ermöglichen das einfache Erstellen von Topologien durch sich wiederholende "Pattern". Durch die übergelagerten APIs sind die Topologien auf einem höheren Level dargestellt und somit leichter zu lesen und zu verstehen. Die Idee ist, dass eine Menge von Objekten (z. B. mehrere Nodes) in einem Container zusammengefasst sind und eine Operation in einem Helper auf einen Container ausgeführt werden kann.

Mithilfe des InternetStackHelper können somit leicht mehreren Nodes alle benötigten Protokolle wie z. B. ARP, IP, TCP, UDP etc. hinzugefügt werden.

3.2. Ausgabemöglichkeiten

Um die Ergebnisse der Simulation später analysieren zu können, stehen in NS-3 mehrere Ausgabe- und Darstellungsmöglichkeiten zur Verfügung.

Für die Analyse sind zwei Arten von Helper-Klassen vorhanden, die *ASCIITraceHelper* und die *PcapHelper*. Mit Hilfe der *ASCIITraceHelper* kann die Kommunikation in einer ASCII Trace-Datei gespeichert werden. Die Kommunikation kann protokollspezifisch (IPv4 & IPv6) oder aber pro NetDevice ausgegeben werden, wie im nachfolgenden Beispiel alle NetDevices, die sich im CSMA Netz befinden.

```
AsciiTraceHelper ascii;
csma.EnableAsciiAll(ascii.CreateFileStream
("ping6.tr"));
```

Die andere Möglichkeit ist die Ausgabe als pcap-Datei durch die *PcapHelper*. Diese Datei kann anschließend zum Beispiel mit Wireshark dargestellt und analysiert werden. Die *PcapHelper* stehen wie bei dem ASCIITraceHelper sowohl den einzelnen NetDevices als auch den Protokollen IPv4 und IPv6 zur Verfügung.

Natürlich können über entsprechende C++ Befehle auch unabhängig von ASCII-Traces und pcap-Dateien benutzerdefinierte Ausgaben erzeugt werden.

Zur visuellen Darstellung kann ab Version 3.10 PyViz verwendet werden. PyViz dient zur Live-Darstellung von Simulationen, wofür keine Trace-Dateien benötigt werden [9].

Eine grafische Eingabe zur Erstellung von Szenarien ist nicht vorhanden. Dies ist weiterhin nur per C++ Programmierung möglich. Mehr zur Erstellung von Szenarien wird in Kapitel 5 beschrieben.

4. IPv6 Unterstützung von NS-3

In diesem Kapitel soll die IPv6-Fähigkeit von NS-3 analysiert werden. Es sollen unter anderem folgende Fragestellungen geklärt werden. Welche Funktionen von IPv6 wurden bis jetzt implementiert? Ist die Übertragung von Schicht-4 Protokollen über IPv6 möglich? Ist ein DualStack umsetzbar? Im nächsten Kapitel wird dann auf ein konkretes Beispielszenario eingegangen.

Obwohl IPv6 standardisiert schon seit 1998 existiert und inzwischen seit einigen Jahren auch praktischen Einsatz beim Endkunden findet, ist in NS-3 keine vollständige Unterstützung zu finden.

Für die IPv6 Unterstützung sind zur Zeit ca. 60 Klassen implementiert. Die Klassen *Ipv6Extension* und *Ipv6ExtensionHeader* bzw. deren Unterklassen sorgen für die Bereitstellung von IPv6-Extensions. Es existieren folgende Implementierungen: *HopByHop, Fragmentation, Authentication Header, Routing, Encapsulating Security Payload (ESP)* und *Destination Options*. Somit sind für alle vorgesehenen IPv6-Erweiterungen im Standard Unterklassen vorhanden. Jumbograms (Pakete mit der Größen von bis zu 4.294.967.335 Byte), können mithilfe der Klasse *Ipv6OptionJumbogram* und *Ipv6ExtensionHopByHop* übertragen werden [6].

Die wesentliche Logik von IPv6 befindet sich in der *IPv6L3Protocol* Klasse [6]. Hier existieren Methoden für den Zugriff auf Routinginformationen, sowie das Senden und Empfangen von Paketen. Bei der Send-Methode, die von höheren Schichten aufgerufen werden kann, werden die Pakete über IPv6 den Stack abwärts an Data- bzw. Physikal-Layer übertragen. Die Receive-Methode wird aufgerufen, wenn ein Paket im Stack eintrifft. Hierbei wird der IPv6-Header entfernt und der Datenteil nach oben an ein Protokoll der Transportschicht gereicht.

Transportschicht

Zur Zeit existiert keine Unterstützung für die Transportschicht (Layer-4). Für diese wäre die abstrakte Klasse *IPv6L4Protocol* vorgesehen [6]. Aber einzig ICMPv6 ist in NS-3 als Unterklasse von *IPv6L4Protocol* verfügbar. Da ICMP aber laut OSI-Referenzmodell der Schicht drei zuzuordnen ist, ist keine Implementierung der Schicht 4 für IPv6 in NS-3 vorhanden.

ICMPv6

ICMPv6 ist die für IPv6 geeignete Version des Internet Control Message Protocols (ICMP). Es verfügt laut RFC 4443 über zwei Typen von Nachrichten. Dies sind die Error-Messages, die sich nochmals in Destination Unreachable, Packet Too Big, Time Exceeded und Parameter Problem unterteilen lassen, sowie die Informational-Messages, die sich wiederum in Request und Reply aufgliedern. All diese sind auch in NS-3 vorhanden [6]. ICMPv6 Requests können am einfachsten über den *Ping6Helper* generiert werden.

Das Neigbour Discovery Protokoll welches ICMPv6 verwendet, ist auch implementiert. Dieses kombiniert das Address Resolution Protokoll und ICMP Router Discovery/Redirect aus IPv4. Zusätzlich bietet es noch weitere Funktionen wie z. B. die Autokonfiguration von IPv6 Adressen und das Verhindern von doppelt vorhandenen Adressen - Duplicate IP address detection (DAD) [1].

Damit ein Node, der als Router definiert ist, auch Router Advertisements versendet bzw. auf Clientanfragen reagiert, muss auf diesem Node zusätzlich noch der "Router Advertisement Daemon" (Klasse *Radvd*) als Applikation eingebunden werden [6].

Adresstypen
Es sind mehrere Adresstypen in NS-3 vorhanden. So können einem Interface Link-Lokale und Globale Adressen zugeteilt werden. Wird die Klasse *Ipv6AddressHelper* zur Adresszuweisung verwendet, wird dem Interface automatisch zuerst eine Link-Lokale Adresse zugewiesen und dann eine Globale. Zur Vermeidung von doppelten IP-Adressen wird hierbei auch das DAD-Verfahren verwendet. Dieses Verfahren überprüft mithilfe eines Multicasts an alle Nodes, ob die IPv6 Adresse schon von einem anderen Node verwendet wird. Dafür sind die spezielle Adresse *::0* (unspecified address) und die Multicast-Adresse *ff02::1* (All-Nodes) erforderlich, die auch in NS-3 implementiert sind. Zudem sind die Lokalhost-Adresse (*::1*) und die AllRouter-Adresse (ff02::2) vorhanden.

Routing
Routingprotokolle für IPv6 existieren in NS-3 nur minimal. Wie bereits erwähnt, wird die IPv6-Erweiterung Routing unterstützt. Hierbei gibt der Source Node im Routing-Header an, welche weiteren Nodes das Paket durchlaufen soll.

Des Weiteren ist statisches Routing für IPv6 mittels der Klasse *Ipv6StaticRouting* möglich. Hiermit können Host-/Netz- und Default-Routen gesetzt werden.

Dual Stack
Die Simulation eines Dual-Stack ist in NS-3 möglich. Falls die Klasse *InternetStackHelper* verwendet wird, sind automatisch der IPv4 und IPv6 Stack auf dem Node aktiv. Dies wird auch noch einmal im nächsten Kapitel 5 "Beispiel Szenario" ersichtlich.

5. Beispiel Szenario

Als Beispiel soll das einfache von NS-3 mitgelieferte Ping6 Szenario betrachtet werden, um einen Einblick in die Erstellung eines Szenarios zu geben und zu zeigen, dass IPv6 lauffähig ist. Das Szenario simuliert einen ICMPv6 Echo Request von Node0 an alle weiteren Nodes im gleichen Netz durch die Multicastadresse ff02::1 und zurück.

Mitgeliefert als Beispielsimulation bezüglich IPv6 wurde von NS-3 in der Version 3.12 neben ICMPv6-Request auch eine Fragmentierung von IPv6 Paketen (fragmentation-ipv6.cc), ICMPv6 Redirekt (icmpv6-redirect.cc), ein Routing Beispiel (loose-routing-ipv6.cc), zwei Router Advertisement Szenarien (radvd.cc & radvd-two-prefix.cc), sowie eine IPv6 Adressgenerierung aus einer MAC-Adresse und einem Netzpräfix (test-ipv6.cc). Ein Tutoriell speziell für IPv6 existiert nicht.

Für die Situation bzw. für die Erstellung eines Szenarios sind mehrere Schritte erforderlich. Abbildung 2 spiegelt einen typischen Ablauf einer Simulation von der Erstellung bis zur Analyse wieder. Da es keine grafische Oberfläche zur Erstellung von Szenarien gibt, müssen diese per C++ erstellt werden. Das Ping6 Szenario soll anhand dieser Schritte dargestellt werden.

Schritt 1: Mithilfe von Helper und Container kann schnell und einfach eine Topologie erstellt werden. In dem Ping6-Szenario wäre dies zum Beispiel die Erstellung von vier Nodes und des zugrundelegenden Netzes durch nachfolgende Befehle:

```
NodeContainer n;
n.Create (4);
CsmaHelper csma;
```

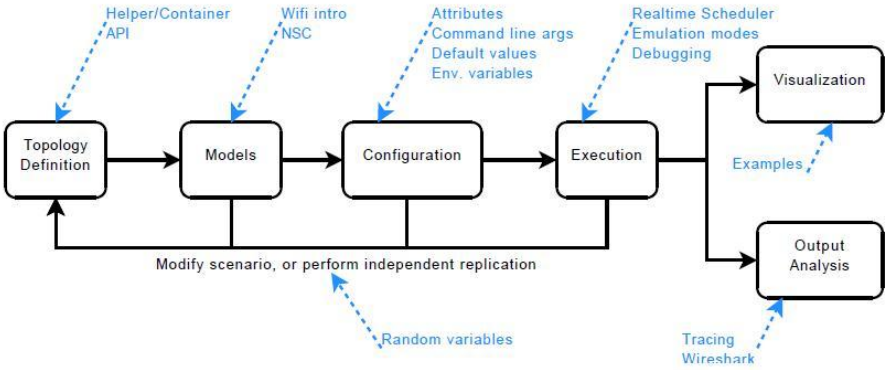

Abbildung 2: Simulationsstruktur [2]

Schritt 2: Weitere einzelne Models, wie z. B. WiFi werden in diesem Beispiel nicht hinzugefügt.

Die eigentliche IPv6-Fähigkeit wird durch den nachfolgenden Befehl erreicht, bei dem der IPv6-Stack eingebunden wird. Zudem wird noch eine IPv6-fähige Applikation zum Senden von ICMPv6-Requests benötigt (siehe Ping6Helper). Mithilfe des InternetStack-Helpers werden alle für IPv6 benötigten Komponenten hinzugefügt. Auch der IPv4-Stack ist standardmäßig aktiviert, in diesem konkreten Beispiel jedoch explizit deaktiviert.

```
InternetStackHelper internetv6;
internetv6.SetIpv4StackInstall (false);
internetv6.Install (n);
```

Schritt 3: Nachdem die Topologie und Models erstellt wurden, können diese konfiguriert werden. In dem Beispiel werden z. B. für die Ping-Applikation das Interface, die Remoteadresse (ff02::1), sowie die Paketeigenschaften gesetzt.

```
Ping6Helper ping6;
ping6.SetIfIndex (i.GetInterfaceIndex(0));
ping6.SetRemote
  (Ipv6Address::GetAllNodesMulticast ());
ping6.SetAttribute ("MaxPackets",
  UintegerValue (maxPacketCount));
ping6.SetAttribute ("Interval", TimeValue
  (interPacketInterval));
ping6.SetAttribute ("PacketSize",
  UintegerValue (packetSize));
```

Schritt 4 & 5:
Danach kann die Simulation mit dem Befehl
`Simulator::Run ();`
gestartet und das Ergebnis anschließend analysiert werden.

Durch einige geringfügige Veränderungen an dem Beispiel kann auch die Unterstützung eines Dual-Stacks gezeigt werden. Hierzu müssen lediglich noch eine Klasse Ipv4AddressHelper (zur IPv4 Adress-generierung) und eine weitere Applikation mittels V4PingHelper (als Ping-Applikation) hinzugefügt und konfiguriert, sowie beim schon vorhandenen InternetStack-Helper die vorher abgeschaltete IPv4-Unterstützung wieder aktiviert werden.

6. Bewertung der IPv6 Unterstützung

Zur Bewertung der IPv6 Unterstützung von NS-3 wird diese verglichen mit dem Umfang der Implementierung von IPv4 in NS-3. Der größte Unterschied besteht darin, dass für IPv6 keine Protokolle der Transportschicht und höherer Schichten existieren, wogegen bei IPv4 verschiedene Implementierungen für TCP (TCP_Reno, TCP_Tahoe) und UDP vorhanden sind. Natürlich können dadurch auch keine Implementierungen von Applikationen, die TCP oder UDP-Pakete über IPv6 versenden, bereitgestellt werden. Für IPv4 existieren dafür zum Beispiel ein UDP-Client und Server, die Pakete senden und empfangen können.

Zusätzlich sind für IPv4 mehr Routingprotokolle vorhanden als für IPv6. So existieren z. B. der Ad-hoc On-demand Distance Vector (AODV) oder Destination-Sequenced Distance-Vector (DSDV) Routingalgorithmus zum Weiterleiten von Daten in kabellosen Ad-hoc-Netzen. Zudem ist die Dokumentation von IPv6 noch nicht so umfangreich und

detailliert wie bei IPv4. Von den genannten Punkten abgesehen ist IPv6 jedoch in NS-3 auf einem ähnlichen Stand wie IPv4.

7. Ausblick

Am 23.12.2011 ist Version 3.13 erschienen, die auch hinsichtlich IPv6 einige Neuerungen bieten sollte. Kurz vor dem vorgesehenen Releasetermin wurden jedoch einige Features gestrichen bzw. auf die Version 3.14 verschoben (voraussichtlicher Releasetermin ist April 2012). So werden selbst in der Version 3.13 noch keine Protokolle auf der Transportschicht über IPv6 unterstützt [8].

Nur eine Funktion zur automatischen IPv6 Adressgenerierung (Ipv6 Address Generator) wurde realisiert. Diese erleichtert das Einrichten von vielen Nodes in einem IPv6 Netz. Der Generator weist den Interfaces sequenziell IPv6-Adressen von einer zur Verfügung gestellten Netzwerkadresse/Maske zu.

8. Fazit

NS-3 ist ein sehr umfangreicher Netzsimulator. Dennoch sind nach einer gewissen Einarbeitungszeit die Erstellung von eigenen Szenarien, deren Simulation und anschließende Analyse problemlos möglich.

Obwohl IPv6 schon seit 1998 existiert, ist es noch nicht vollständig in NS-3 integriert. Auch fehlt es zur Zeit noch an geeigneter Dokumentation bzgl. IPv6. Es ist davon auszugehen, dass diese aktualisiert und neue IPv6 Funktionen implementiert werden. Am Beispiel der aktuellen Version, die am 23.12.2011 erschienen ist, wird jedoch ersichtlich, dass hierbei häufig Verzögerungen auftreten können. Wann eine vollständige IPv6 Implementierung vorhanden und adäquat dokumentiert sein wird, und somit IPv6-Szenarien umfangreich simuliert werden können, ist zu diesem Zeitpunkt noch nicht genau absehbar.

9. Quellen

[1] Hagen, S.: “IPv6 Essentials (2. Auflage)“; Sebastopol: O'Reilly 2006.

[2] Lacage, M: “Experimentation with ns-3”; http://www.nsnam.org/tutorials/trilogy-summer-school.pdf (abgerufen am 12.01.2012)

[3] Lacage, M: “An ns-3 tutorial“; http://www.nsnam.org/tutorials/ns-3-tutorial-tunis-apr09.pdf (abgerufen am 12.01.2012)

[4] Montavont, N & J; Vincent, S.: Implementation of an IPv6 Stack for NS-3 http://www.wns2.org/docs/paper9250-Implementation%20of%20an%20IPv6%20Stack%20for%20NS-3.pdf (abgerufen am 18.12.2011).

[5] NS-3. http://www.nsnam.org (abgerufen am 18.12.2011).

[6] NS-3: "Doxygen"; http://www.nsnam.org/docs/release/3.12/doxygen/annotated.html (abgerufen am 18.12.2011)

[7] NS-3: "ns-2 & ns-3"; http://www.nsnam.org/support/faq/ns2-ns3/ (abgerufen am 18.12.2011).

[8] NS-3: "Ns-3.13"; http://www.nsnam.org/wiki/index.php/Ns-3.13 (abgerufen am 18.12.2011).

[9] NS-3: "PyViz"; http://www.nsnam.org/wiki/index.php/PyViz (abgerufen am 18.12.2011).