

**Humboldt-Universität zu Berlin**

**Mathematisch-naturwissenschaftliche Fakultät II**

**Institut für Informatik**



**Simulation eines Netzwerkes mit ns2,  
indem die Knoten Restart durchführen**

**Halbkurs: Modellbasierte Leistungs-  
und Zuverlässigkeitsanalyse**

**Semester: Wintersemester 06/07**

**Dozentin: Dr. Katinka Wolter**

Willi Engel,

Alexandra Gelemerova,

# Inhaltsverzeichnis

<b>1 Einführung.....</b>	<b>3</b>
<b>2 NS2 – Network Simulator.....</b>	<b>4</b>
2.1 Einleitung.....	4
2.2 Geschichte.....	4
2.3 Charakteristiken.....	4
2.4 Programmablauf von ns2.....	5
2.4.1 Bestandteile von ns2.....	6
2.4.1.1 Tcl/Tk .....	7
2.4.1.2 OTcl .....	7
2.4.1.3 TclCL.....	7
2.4.1.4 ns-2 .....	7
2.4.1.5 Nam-1 .....	7
2.4.1.6 Xgraph.....	8
2.5 Installation.....	8
2.5.1 Installation im Rahmen unserer Netzwerksimulation.....	8
<b>3 Netzwerksimulation mit ns2.....</b>	<b>10</b>
3.1 Netzwerktopologie.....	10
3.2 Restart-Algorithmus.....	12
3.3 Protokollieren der Ergebnisse.....	13
3.4 Analyse der Ergebnissen.....	15
3.5 Übertragungsrate 0,3 Mb pro Route.....	17
3.5.1 Übertragungsrate 0,3 Mb und Packetgröße 1000Byte.....	17
3.5.2 Übertragungsrate 0,3 Mb und Packetgröße 100Byte.....	19
3.6 Übertragungsrate 0,4 Mb pro Link.....	22
3.6.1 Übertragungsrate 0,4 Mb und Packetgröße 1000Byte.....	22
3.6.2 Übertragungsrate 0,4 Mb und Packetgröße 100Byte.....	25
3.7 Übertragungsrate 1,0 Mb pro Link.....	28
3.7.1 Übertragungsrate 1,0 Mb und Packetgröße 1000Byte.....	28
3.7.2 Übertragungsrate 1,0 Mb und Packetgröße 100Byte.....	30
3.8 Zusammenfassung:.....	33

# 1 Einführung

In unserem Projekt benutzen wir das Simulationsprogramm ns2, um ein TCP-Netzwerk zu definieren und zu simulieren. Dabei wenden wir den Restart-Algorithmus an, um festzustellen, ob der Restart im Falle der Simulation zu besseren Ergebnissen führt oder nicht.

# 2 NS2 – Network Simulator

## 2.1 Einleitung

Der Network Simulator – ns2 ist ein Programm, das Netzwerktopologien darstellt und simuliert. Man kann es als Open Source von der ns2-Seite:

[http://nslam.isi.edu/nslam/index.php/Main\\_Page](http://nslam.isi.edu/nslam/index.php/Main_Page)

herunterladen. Ns2 unterstützt die Simulation verschiedener Protokolle wie TCP, Routing und Multicast-Protokolle über verdrahtete und kabellose Verbindungen, wie WLAN und Satellit<sup>1</sup>.

## 2.2 Geschichte

Ns2 wurde am Anfang als Teil des REAL Network Simulators an der UC Berkeley im Jahre 1989 entwickelt. Es sollte zur Verbesserung von paketorientierten Netzen eingesetzt werden, um Stau- und Lastsituationen zu vermeiden. Am Network Simulator wurde weiter entwickelt und im Jahre 1995 erschien die erste Version ns-1. Das Projekt, unter dem Namen VINT – Virtual InterNetwork Testbed<sup>2</sup>, wurde von einer Reihe von Instituten unterstützt, wie z. B. Defense Advanced Research Projects Agency (DARPA), Lawrence Berkeley National Laboratory (LBNL), Xerox Palo Alto Research Center (PARC), Information Science Institut at University of Southern California (USC/ISI) und der University of California, Berkeley (UCB). Im Jahre 1997 erschien die zweite Version: ns-2, die erweitert und mit noch mehr Simulationsmöglichkeiten verbessert wurde, wie z. B. Real Time Protocol (RTP), Scheduling-Algorithmen und mobile Hosts.<sup>3</sup>

---

<sup>1</sup> Andreas Mosig: Net Simulator 2, Routing Seminar, Universität Koblenz-Landau, 2004, S. 1

<sup>2</sup> Seite von VINT-Projekt: <http://www.isi.edu/nslam/vint/>

<sup>3</sup> Andreas Mosig: Net Simulator 2, Routing Seminar, Universität Koblenz-Landau, 2004, S. 1

## 2.3 Charakteristiken

Der NS-Simulator basiert auf zwei Programmiersprachen: C++ und OTcl. Die zwei Sprachen werden kombiniert, da ns2 zwei verschiedene Aufgaben zu erfüllen hat. Einerseits müssen detaillierte Protokoll-Simulationen realisiert werden. Dieser Prozess braucht eine Programmiersprache wie C++, die mit Paketen und Algorithmen, die viel Daten enthalten, umgehen kann. Andererseits muss man beim Erforschen von Netzwerken schnell Parameter und Konfigurationen verändern können, um verschiedene Fälle zu untersuchen. Deshalb ist es wichtig, dass, nachdem das Modell umkonfiguriert wurde, die Simulation sofort wieder gestartet werden kann. Die erste Aufgabe wird C++ implementiert. Die Schicht von C++ hilft dabei, die Effizienz der Simulation zu steigern und die Größe der Ausführungsdateien zu verringern. Durch OTcl kann man schnell eine Konfiguration ändern und die Simulation neu starten, was diese Sprache sehr bequem für das Erstellen von einzelnen Simulationen macht.<sup>4</sup>

Ein Netzwerk wird mit Hilfe der Skriptsprache Tcl beschrieben. Die Tcl-Datei wird dann in der Eingabekonzole durch das Kommando `ns <Dateiname>` aufgerufen. Durch die Visualisierungstools Nam und XGraph kann man im Anschluß die Simulation des Netzwerks graphisch beobachten. Der Nam-Editor stellt die Simulation in Form einer Animation dar und XGraph zeigt die Ergebnisse als Graphik, in der die Ausführungs- und die Antwortzeiten durch zwei Koordinaten X und Y abgelesen werden können. Ns2 erstellt automatisch eine Trace-Datei, wohin die Ergebnisse der Simulation umgeleitet werden. Auf diese Weise wird jedes Ereignis vom Ausführen des Netzwerks dokumentiert. Die Simulation findet als interne Berechnung statt.

---

<sup>4</sup> Eitan Altman, Tania Jiménez: NS Simulator for Beginners; Univ. de Los Andes, Mérida, Venezuela and ESSI, 2003-2004, S. 8

## 2.4 Programmablauf von ns2

Die Skriptdatei, in OTcl geschrieben und vom Nutzer programmiert, enthält die Netzwerktopologie, die verwendeten Protokolle und die Routingmechanismen, wie auch Start- und Endzeiten, Geschwindigkeit und verschiedene Ereignisse, die im Lauf der Simulation auftreten können, z. B. Ausfall einer Verbindung oder erneutes Senden von Informationen nach einem bestimmten Zeitintervall. Der Network Simulator verarbeitet die Daten, die das Programm abliefert, und berechnet die Ergebnisse. Zuerst werden die verschiedenen Knoten konfiguriert, unter denen gibt es Sender, Router und Receiver. Je zwei Knoten werden durch Links verbunden. Die Links können simplex- oder duplex-link sein, abhängig davon, ob Information nur in eine oder in beide Richtungen fließen. Dabei besteht die duplex-Verbindung aus zwei simplex-Links. Wenn man in Hin- und Rückrichtung verschiedene Datenflüsse erzeugen will, dann muss man zwei simplex-links erstellen.<sup>5</sup> Jedem Knoten wird ein Agent zugeordnet. Die Agents tragen die Protokollfunktionen (z. B. TCP, UDP) und sind mit den entsprechenden Senken (Receivern) verbunden. Der Datenverkehr ‚traffic‘ wird über die Agents geleitet, läuft durch Knoten und gelangt in die Senken<sup>6</sup>. Zu bestimmten Zeitpunkten werden Ereignisse ‚events‘ produziert.<sup>7</sup> Ein Event ist das Eintreffen bzw. Senden eines Datenpaketes in einem Knoten. Zusätzlich wird über events auch das Verhalten der Simulation aus OTcl gesteuert, zum Beispiel wird mit Hilfe eines Events das Starten des traffics ausgelöst.

### 2.4.1 Bestandteile von ns2

Ns2 schließt mehrere Bestandteile ein: Tcl/Tk, OTcl, TclCL, ns-2, nam-1, XGraph, Perl, Tcl-debug, Dmalloc, Sgb2ns conversion program, Tiers2ns conversion program, Cweb und sgb source code. Die meisten Komponenten sind im Linux-Betriebssystem

---

<sup>5</sup> Chrisian Hofmann, Christoph Lumme:Praktikum Netzwerksimulator 2, Tutorial; Im Rahmen des Projektes SIMON (Simulation Environment for Mobile Networks[3]), Technische Universität Ilmenau, 2005, S. 5

<sup>6</sup> Ebd., S. 3

<sup>7</sup> Andreas Mosig: Net Simulator 2, Routing Seminar, Universität Koblenz-Landau, 2004, S. 8

schon enthalten. Damit man sie alle nacheinander nicht kompilieren muss, kann man auch das all-in-one-package installieren. Das ganze Paket benötigt aber mehr Speicherplatz, als die einzeln konfigurierten Teile. Einige ,für die Simulation wichtigen, Bestandteile sind<sup>8</sup>:

#### *2.4.1.1 Tcl/Tk*

Tool command language/Toolkit: Tcl/Tk ist eine Open-Source-Skriptsprache für die Ausführung von graphischen Oberflächen. In dieser Sprache werden die Anweisungen (Quellcodes) für die Simulation geschrieben. Tcl ist normalerweise schon in Linux vorinstalliert und hat eine C-ähnliche Syntax. Durch das Toolkit kann man graphische Benutzeroberflächen programmieren.

#### *2.4.1.2 OTcl*

Object Tcl: Tcl wird objektorientiert. Durch Otcl kann man mehrere Instanzen von Objekten erstellen.

#### *2.4.1.3 TclCL*

Tcl mit Classes: Dieses Element verbindet die Objekte, die in der Skriptdatei beschrieben wurden, mit ihren Realisierungen in C++.

#### *2.4.1.4 ns-2*

Network simulator: ns2 ist der Hauptteil des ganzen Programms, da es die Daten vom Steuerungsskript berechnet und die Ergebnisse von der Simulation in eine Trace-Datei schreibt.

#### *2.4.1.5 Nam-1*

Network animator: nam-1 ist eine Software, die in Tcl/Tk geschrieben wurde. Sie zeigt die Ergebnisse von den Trace-Dateien. Die Simulation kann auf diese Weise graphisch verfolgt und gesteuert werden. Die

---

<sup>8</sup> Ebd., S. 5

Animation lässt sich zurücksetzen, vorspulen oder anhalten.<sup>9</sup> Man kann mit dem eingebauten Editor eine vorhandene Netzwerktopologie ändern oder eine neue erstellen, sie dann als tcl-Datei speichern und wieder simulieren.

#### *2.4.1.6 Xgraph*

Dieses Tool stellt Dateien graphisch dar, welche als Inhalt x, y- Paare beinhaltet. Mit Xgraph kann man zum Beispiel die Auslastung des Netzwerks zu bestimmten Zeitpunkten sehen.

### *2.5 Installation*

Die neueste Version des Network Simulator ist ns-2.30, freigegeben im September 2006<sup>10</sup>. Ns2 wurde für Unix-Maschinen entwickelt, deshalb läuft er auf Linux, SunOS, Solaris. Es ist möglich ns2 auch unter Windows zu benutzen, dafür benötigt man das zusätzliche Tool cygwin (eine Linux-Umgebung für Windows). Von der offiziellen Seite von cygwin: <http://www.cygwin.com/> kann ist diese Software kostenlos herunterladbar. In Linux kann man ns2 auf zwei Weisen installieren: jeden Bestandteil von ns2 einzeln oder das gesamte Paket, das alle Bestandteile enthält. Wenn man Speicherplatz von mehr als 250 MB auf der Festplatte sparen will, oder Probleme mit dem all-in-one-package hat, dann sollte man die Komponenten nacheinander herunterladen und konfigurieren. Mit dem all-in-one-package ist man aber sicher, dass alle Teile schon enthalten sind und die Installation ist besonders für Neueinsteiger günstig. Mit Cygwin unter Windows funktioniert nur das gesamte Paket von ns2. Informationen über die Konfiguration des all-in-one-package stehen auf der Seite:

[http://nslam.isi.edu/nslam/index.php/Downloading\\_and\\_installing\\_ns-2](http://nslam.isi.edu/nslam/index.php/Downloading_and_installing_ns-2)

---

<sup>9</sup> Chrsian Hofmann, Christoph Lumme:Praktikum Netzwerksimulator 2, Tutorial; Im Rahmen des Projektes SIMON (Simulation Environment for Mobile Networks[3]), Technische Universität Ilmenau, 2005, S. 2

<sup>10</sup> <http://mailman.isi.edu/pipermail/ns-users/2006-September/057343.html>

### 2.5.1 Installation im Rahmen unserer Netzwerksimulation

Um den Restart-Algorithmus in der Simulation zu implementieren, wurde er in C++ umgesetzt. Die RestartAlgorithm.cc-Datei enthält die nötigen Komponenten und wird im Ordner ‚tcp‘ von ns gespeichert. Der Ordner hat normalerweise den folgenden Pfad: `~ns/tcp`<sup>11</sup>. Im selben Ordner werden auch die Dateien tcp.cc und tcp.h ersetzt. Die neuen Dateien tcp.cc bzw. tcp.h berechnen die Antwortzeit der Pakete und übergeben diese dem Restart- Algorithmus. Er optimiert basierend auf den Antwortzeiten den Retransmission Timer. Die Verwendung der neuen Funktionalität ist detailliert in ns\_doc-extension.pdf beschrieben, welche dem separatem Quellcode- Archiv beiliegt. Im Ordner `~ns-2.30/tcl/lib` wird die Datei ns-default.tcl ersetzt, um bestimmte Default-Werte in den tcp.cc Dateien zu setzen.

Die geänderten Dateien werden erst nach einer erneuten Übersetzung von ns2 übernommen. Mit Hilfe von ‚make clean‘ und ‚make‘ kann ns2 neu kompiliert und gelinkt werden. Eine genaue Installationsanleitung ist der Datei ‚readme‘ zu entnehmen, welche im separatem Quellcode-Archiv vorhanden ist.

---

<sup>11</sup>~ns bezeichnet den Ordner in welchem sich ns2 in der aktuellen Version befindet

# 3 Netzwerksimulation mit ns2

Im folgenden soll auf die Simulation eines Netzwerkes eingegangen werden. Ziel dieser Untersuchung ist das Verhalten des Restart- Algorithmuses zu untersuchen. Zu erst wird das der Analyse zu Grunde liegende Netzwerk beschrieben. Im weiteren Verlauf wird auf die Simulation und die Ergebnisse eingegangen.

## 3.1 Netzwerktopologie

Die Netzwerktopologie bestimmt, wie die verschiedenen Knoten zueinander geordnet sind. Die Knoten stellen einzelne Rechner, Server oder Clients<sup>12</sup> dar, die alle im Netz zusammen geschlossen sind. Um die Graphik besser nachzuvollziehen, kann man jedem Knoten und jedem Link einen Namen und Farbe geben<sup>13</sup>. Die Topologie schließt auch die Verbindung zwischen den Knoten ein.

In unserem Netzwerk sind fünf Knoten eingerichtet. Knoten 2, 3 und 4 sind die Sender, Knoten 1 ist der Router und Knoten 0 ist der Empfänger der Daten.

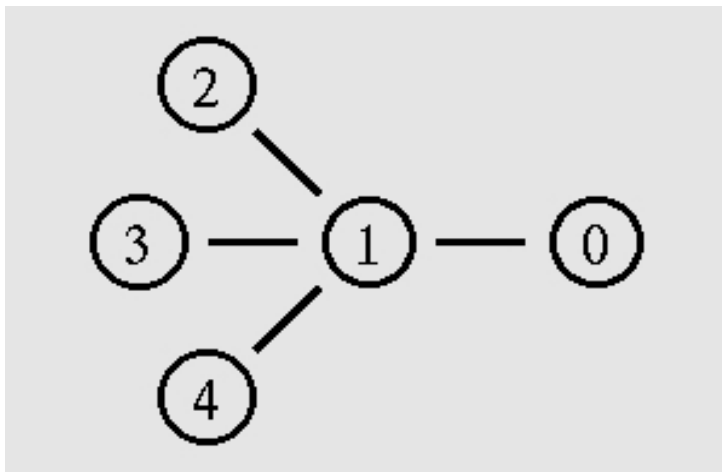


Abbildung 1: Netztopologie in Nam

Man kann die Position der Knoten in der Skriptdatei (\*.tcl) wie folgt bestimmen:

```
$ns duplex-link-op $n1 $n0 orient right
```

```
$ns duplex-link-op $n2 $n1 orient right-down
```

```
$ns duplex-link-op $n3 $n1 orient right
```

<sup>12</sup> Andreas Mosig: Net Simulator 2, Routing Seminar, Universität Koblenz-Landau, 2004, S. 9

<sup>13</sup> Andreas Mosig: Net Simulator 2, Routing Seminar, Universität Koblenz-Landau, 2004, S. 9

ns duplex-link-op ns4 ns1 orient right-up

In Abbildung 2 kann man sehen, dass die Knoten 2, 3 und 4 Pakete zu Knoten 0 senden, die alle durch Knoten 1 laufen. Um die Datenflüsse besser zu beobachten, haben wir grüne, gelbe und rote Farben für die verschiedenen Senderknoten gewählt. Die kleinen Striche, die zwischen 0 und 1 und zwischen 1 und 3 in der Visualisierung mit Nam zu sehen sind, sind die Acknowledgements, die der Empfänger zu den Sendern schickt, um zu bestätigen, dass er die Datenpakete erhalten hat. Jedes Paket hat eine eindeutige Sequenz- Nummer, so erkennen Sender, wie auch Empfänger um welches Pakete es sich handelt. Die Sequenz- Nummer ist eindeutig pro Route. Darüberhinaus hat jedes in ns2 erzeugte Paket eine eindeute ID, welche in der gesamten Simulation nur einmal vorhanden ist. Zur Identifikation der Bestätigungspakete wird die Sequenz- Nummer verwendet. Der Datenstrom der zwischen 1 und 0 nach oben gerichtet ist, repräsentiert die Pakete, in der Warteschlange des Routers (Knoten 1).

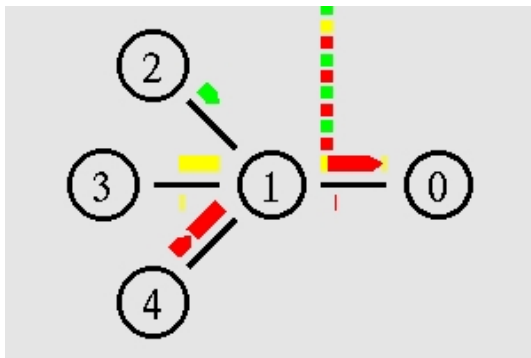


Abbildung 2: Netz mit Last in Nam

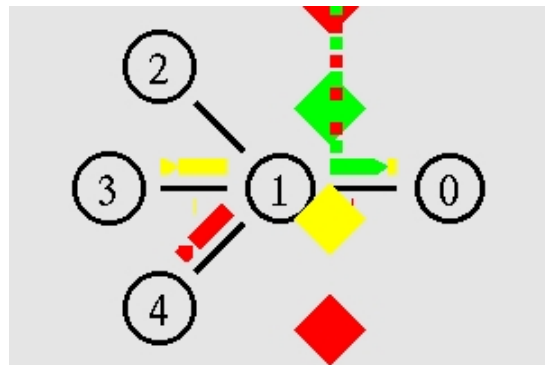


Abbildung 3: Netz mit Überlast in Nam

In Abbildung 3 kann man sehen, wie die Pakete weggeworfen werden und somit verloren sind. Das sind die Pakete, welche in horizontaler Ausrichtung vergrößert dargestellt sind.

### ***3.2 Restart-Algorithmus***

Der Restart wird benutzt, um Aufgaben auszuführen, bei denen Fehler aufgetreten sind, oder wenn diese Aufgaben zu lange Zeit brauchen. Der Restart-Mechanismus kann sehr nützlich beim Protokoll-Verkehr und Internet sein. Restart bedeutet, dass wenn die Antwortzeit für bestimmte Anwendung ein maximales Zeitintervall überspringt, wird ein erneuter Versuch vorgenommen.<sup>14</sup> In den meisten Fällen auf diese Weise wird die Anwendung schneller ausgeführt. Das Betätigen der Reload-Taste in einem Browser ist ein Restart- Algorithmus nach demselben Prinzip. Eine Seite lädt sich zu langsam und der Nutzer drückt auf die Reload-Taste, so dass das alte Laden der Seite abgebrochen wird und ein neues begonnen wird.<sup>15</sup>

In unserem Fall des TCP-Transport wird Restart benutzt, um das Senden und das Empfangen von Paketen zu optimieren. Wenn der Sender nach einer bestimmten Zeit kein Acknowledgement bekommt, so sendet er die selben Pakete erneut. Ein solcher Restart- Timer ist bereits im Standard von TCP vorgesehen. Die Werte dieses Retransmission- Timers werden jedoch statisch vorgegeben. Die Berechnungsvorschrift wie, sie in ns2 Implementiert ist, lautet:

$$\text{neuerRetransmissionTimer}=2*\text{alterRetransmissionTimer}.$$

Der neue Algorithmus berechnet nun den Retransmission- Timer in Abhängigkeit der Antwortzeiten bisheriger Pakete dynamisch. Eine Anpassung an Netzgegebenheiten und daher eine Verbesserung der Netzcharakteristik ist die erhoffte Folge.

### ***3.3 Protokollieren der Ergebnisse***

Um die Simulation zu realisieren, haben wir drei Skriptdateien erstellt. Die eine definiert das Netzwerk, die Topologie, die Protokolle, die Verbindungen und die

---

<sup>14</sup> Katinka Wolter, Philipp Reinecke, Aad van Moorsel: A Measurement Study of the Interplay Between, Application Level Restart and Transport, Protocol, Humboldt-Universität zu Berlin, Institut für Informatik S. 1

<sup>15</sup> Katinka Wolter, Aad P. A. van Moorsel: Analysis and Algorithms for Restart; Humboldt-Universität zu Berlin, Institut für Informatik; University of Newcastle, School of Computing Science, S. 1

Ereignisse. Die andere Datei ist als Package programmiert und erstellt die Dateien für die Xgraph- Ausgaben und berechnet einige Statistiken. Die dritte ist eine Indexdatei, da in Tcl die Packages ein zusätzliches Indexfile brauchen. Wir haben drei Output-Dateien (richtigesNetz.nam, richtigesNetz.tra, richtigesNetz.own), die die Ereignisse der Simulation dokumentieren. Das sind trace-files, wo in ascii-Form in mehreren Spalten vom Programm automatisch verschiedene Informationen eingetragen werden, wie z. B. Art des Ereignisses, Paketgröße, Start- und Endknoten, Start- und Endzeit, FlowID usw. Die ersten beiden Dateien \*.nam und \*.tra werden in der Originalversion von ns2 erzeugt. Das in der der Datei \*.own verwendete Format wurde speziell zur Aufzeichnung der Antwortzeiten und Retransmission- Timer von uns in den tcp.cc/.h- Dateien erzeugt.

Am Ende der Simulation (wenn die tcl- Dateien abgearbeitet sind) erscheint in der Eingabekonsole eine Informationsmeldung, die die Zahl der gesendeten und verloren gegangenen Pakete enthält, wie auch die Zahl der Pakete, für die ein Acknowledgement bekommen wurde. Die Zahl der Acknowledgements, die gesendet wurden und die, die verloren gegangen sind, werden ebenfalls berechnet und gezeigt (siehe: Abbildung 4). Darüberhinaus befinden sich mehrere Dateien mit der Endung .xgr im Simulationsverzeichnis. Diese Dateien enthalten die x-y- Paare für die Xgraph- Ausgaben. Für jeden Sendeknoten wurde eine Datei für die Antwortzeit und eine für die Werte des Retransmission- Timer erstellt. Die Zuordnung der Dateien zu den Knoten ist durch die Knoten- ID im Dateinamen erkennbar.

```
 /usr/local/projekt
$ cd projekt
Alex@PC260949806221 /usr/local/projekt
$ export DISPLAY=:0.0
Alex@PC260949806221 /usr/local/projekt
$ ns richtigesNetz.tcl

==== Some statistiks ====
Number of packages sent (without ack-pck): 3074
Number of acknowledge packages sent: 939
Number of packages dropped (without ack-pck): 2135
Number of acknowledge packages dropped: 0

Number of acknowledged packages: 74

starting xgraph - `xgraph -M -nl -t "Response time" -x "time of the first packag
e sent in sec" -y "response time in sec" response_node_2.xgr response_node_3.xgr
response_node_4.xgr `
starting xgraph - `xgraph -M -t "Retransmission Timer" -x "time in sec" -y "Rtx
Timer value in sec" rtx_node_3.xgr rtx_node_4.xgr rtx_node_2.xgr `

Alex@PC260949806221 /usr/local/projekt
$ Cannot connect to existing nam instance. Starting a new one...
```

Abbildung 4: Cygwin Ausgabe

### 3.4 Analyse der Ergebnissen

Im folgenden werden die Ergebnisse verschiedener Simulationen dargestellt und analysiert. Allen Simulationen liegt die folgende Topologie zu Grunde.

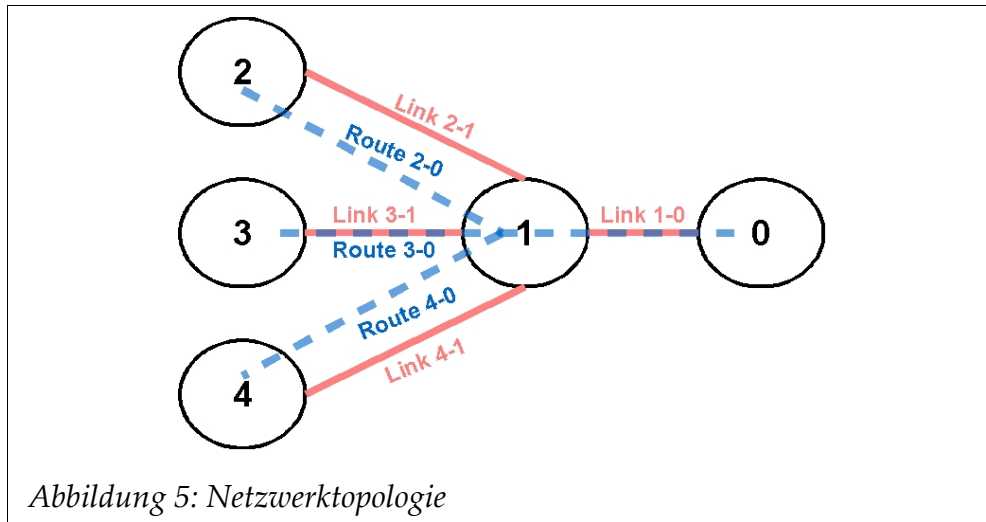


Abbildung 5: Netzwerktopologie

Jeder Link hat eine maximale Kapazität von 1Mb. Zusätzlich wurden drei Routen eingerichtet, welche den Verkehr von Knoten 2 zu 0, von Knoten 3 zu 0 und von Knoten 4 zu 0 bewerkstelligen. Durch diese Konfiguration wird der Link 1-0 zum Flaschenhals. Für die Übertragung wird das TCP- Protokoll gewählt. Auf der TCP-Schicht setzt eine Applikation auf, welche eine konstante Bitrate auf den Routen erzeugt. In den Simulationen werden allen Routen die gleichen Übertragungsraten zugeteilt, wobei sich die einzel Raten auf Link 1 – 0 additiv überlagern. Für den Knoten 1 wurde eine faire Warteschlange verwendet, somit ist jeder Knoten in der Warteschlange gleichberechtigt. Solch ein Verhalten ist im allgemeinen nicht konform zu gängigen Routern, wird aber verwendet um etwaige Dominanzen von Knoten außen vor zu lassen. Alle Simulationen haben eine Simulationszeit von 30 Sekunden.

Für die verschiedenen Simulationen werden die Übertragungsleistung der Applikationsschicht verändert, so wie die Größe der Übertragungspakete (auf TCP- und Applikationsebene).

Im nachfolgenden werden die Ergebnisse der Simulationen unter Verwendung des Restart- Algorithmuses und ohne den Algorithmus gegenübergestellt. Hierzu werden als erstes die Ergebnisse für die Antwortzeit der Pakete und die Werte des Retransmission- Timers grafisch dargestellt. Anschließend erfolgt die Auswertung unter den Gesichtspunkten:

1. Verhalten des Netzes mit Restart vs. Verhalten ohne Restart
2. Einfluß der Paketgröße

In Grafiken bezüglich der Antwortzeit ist auf der X-Achse die Simulationszeit abgetragen. Auf der Y- Achse wird die Antwortzeit gemessen. Für jedes bestätigte Pakete wird ein Punkt im Koordinatensystem erzeugt mit der Zeit zu welcher das Paket zum ersten Mal gesendet wurde als x- Parameter. Der y- Wert dieses Paketes bezeichnet die Antwortzeit. Die Antwortzeit ergibt sich aus: Zeitpunkt der Ankunft des Bestätigungspaketes - Zeitpunkt des Starts der ersten Übertragung des Paketes. Die X- Achse in den Retransmission - („Minus“) Timer Abbildungen zeigt die Simulationszeit. Die Y- Achse den Wert des Timers zu dem Zeitpunkt der Simulation.

### 3.5 Übertragungsrate 0,3 Mb pro Route

#### 3.5.1 Übertragungsrate 0,3 Mb und Paketgröße 1000Byte

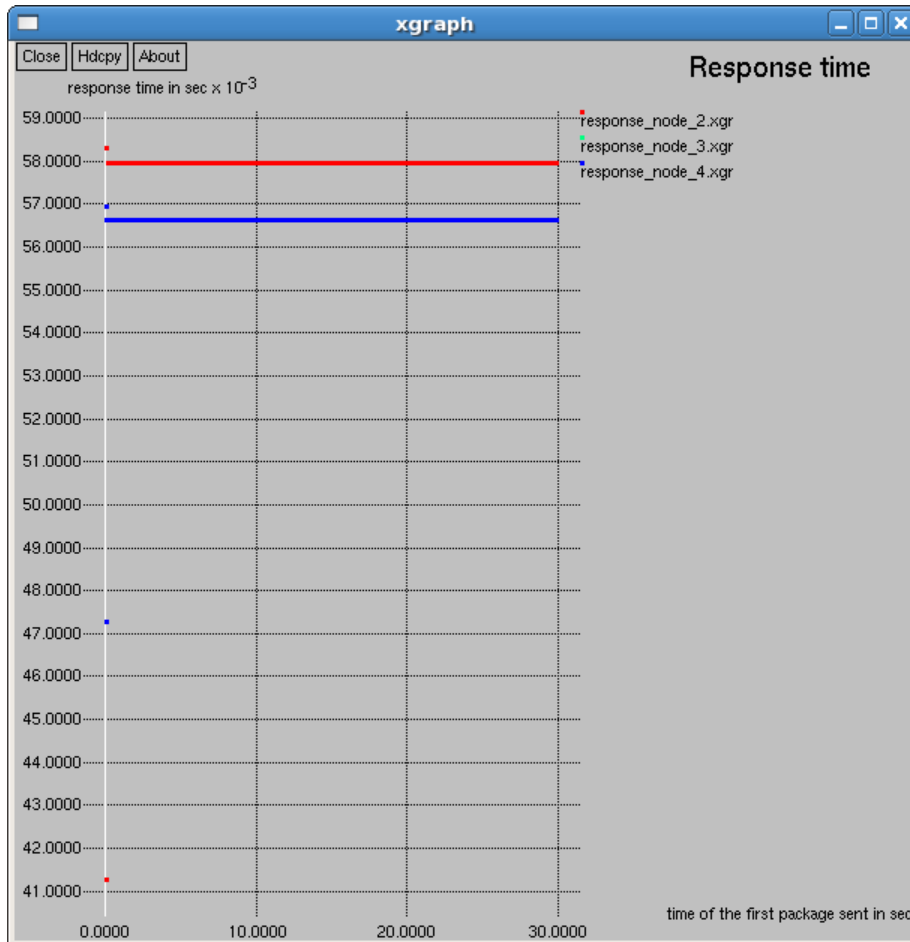


Abbildung 6: Antwortzeit mit Restart

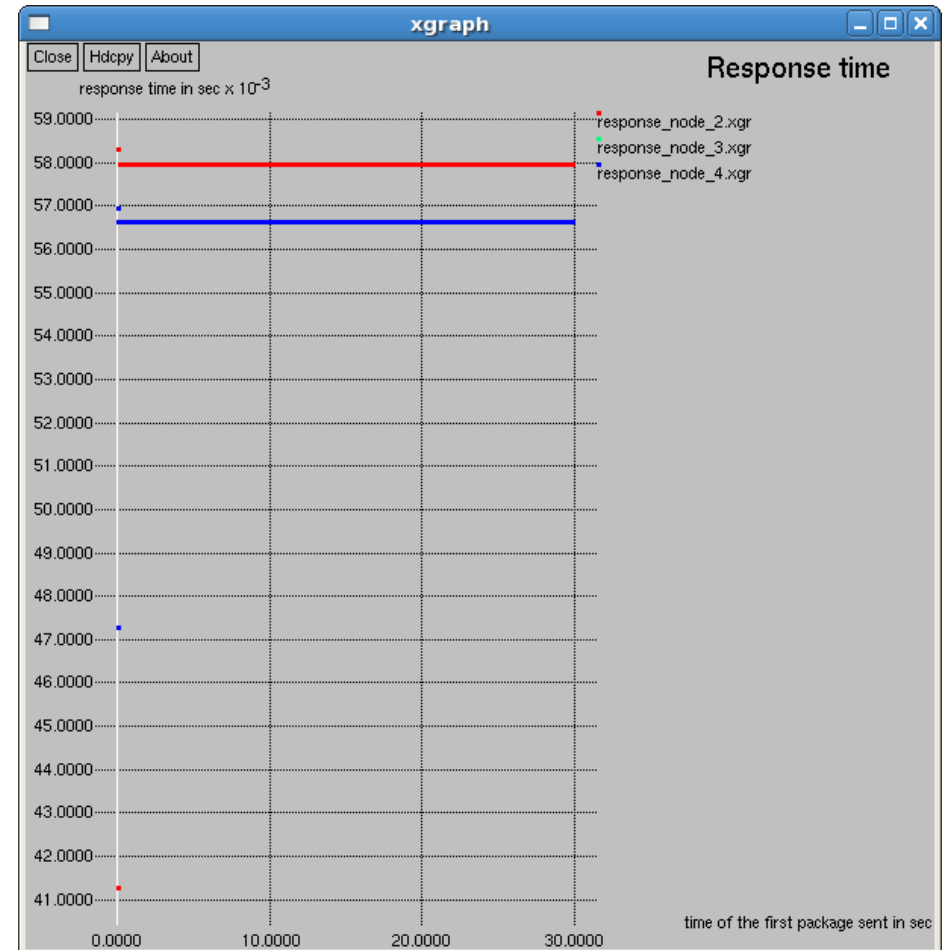


Abbildung 7: Antwortzeit ohne Restart

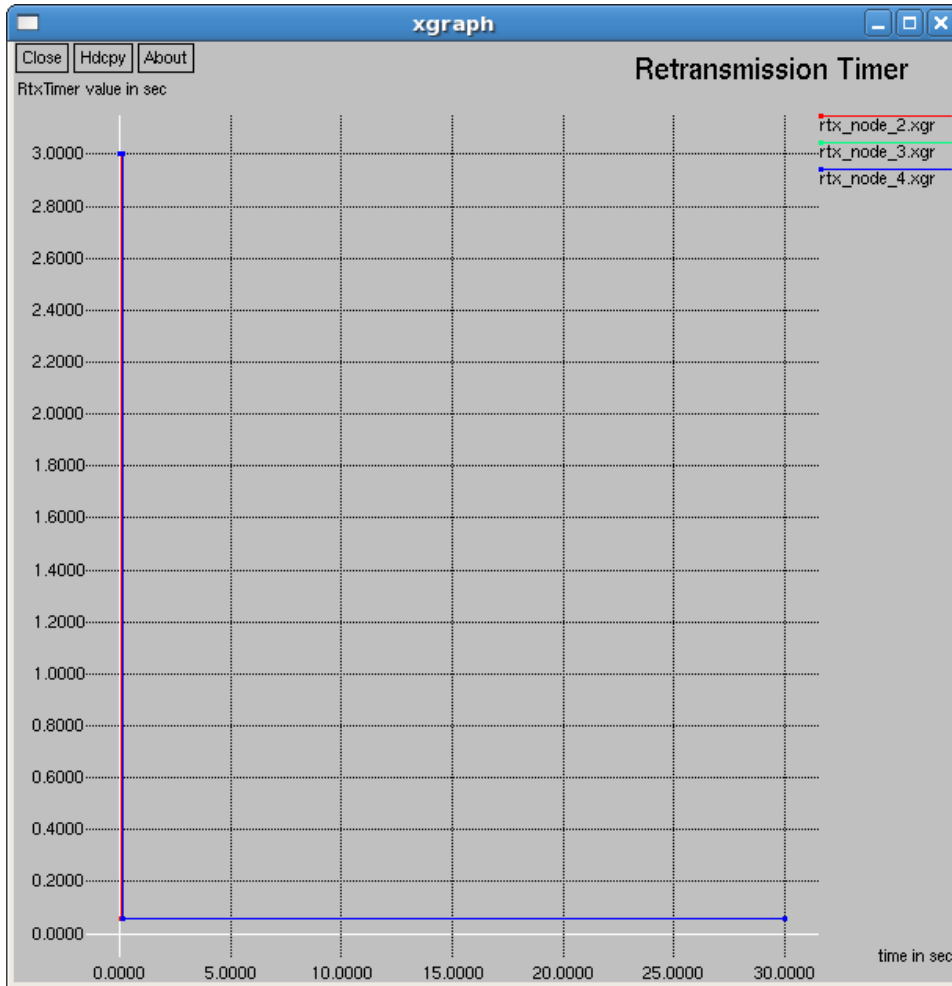


Abbildung 8: Retransmission- Timer mit Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of packages sent (without ack-pck)	3373	3373
Number of acknowledge packages sent:	3369	3369
Number of packages dropped (without ack-pck)	0	0

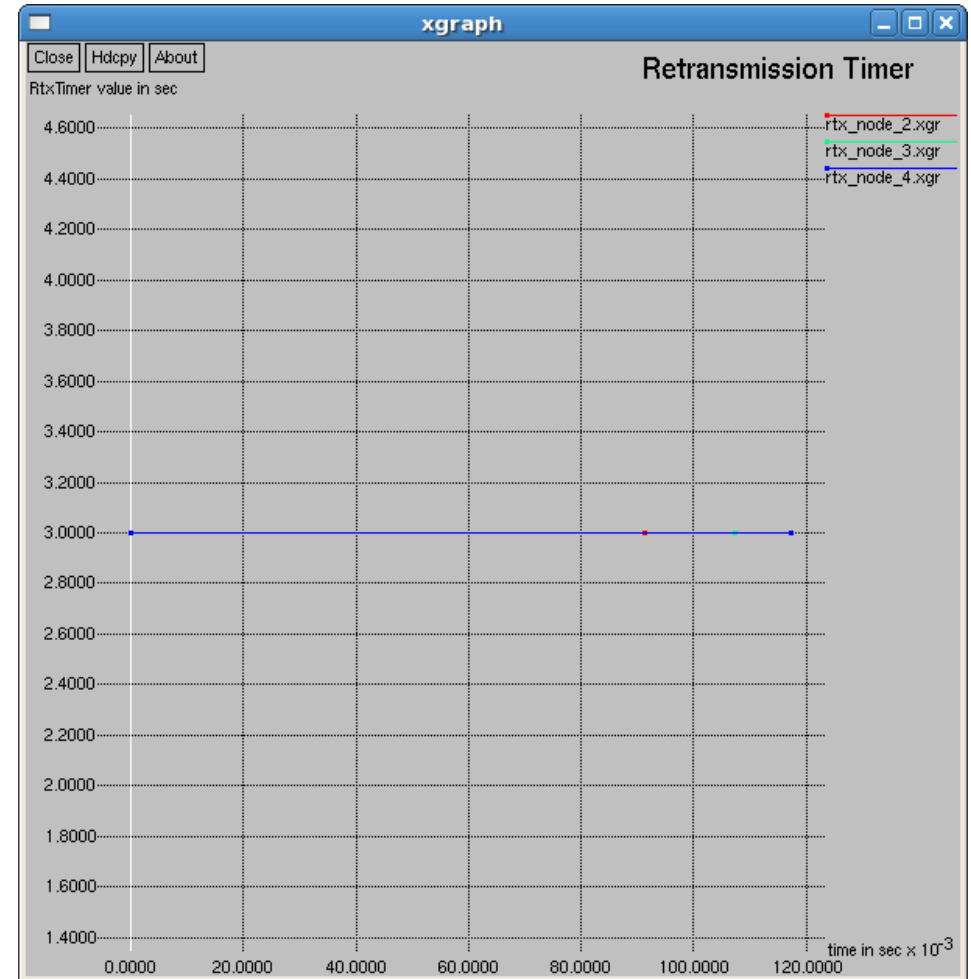


Abbildung 9: Retransmission- Timer ohne Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of acknowledge packages dropped	0	0
Number of acknowledged packages	3366	3366

### 3.5.2 Übertragungsrate 0,3 Mb und Packetgröße 100Byte

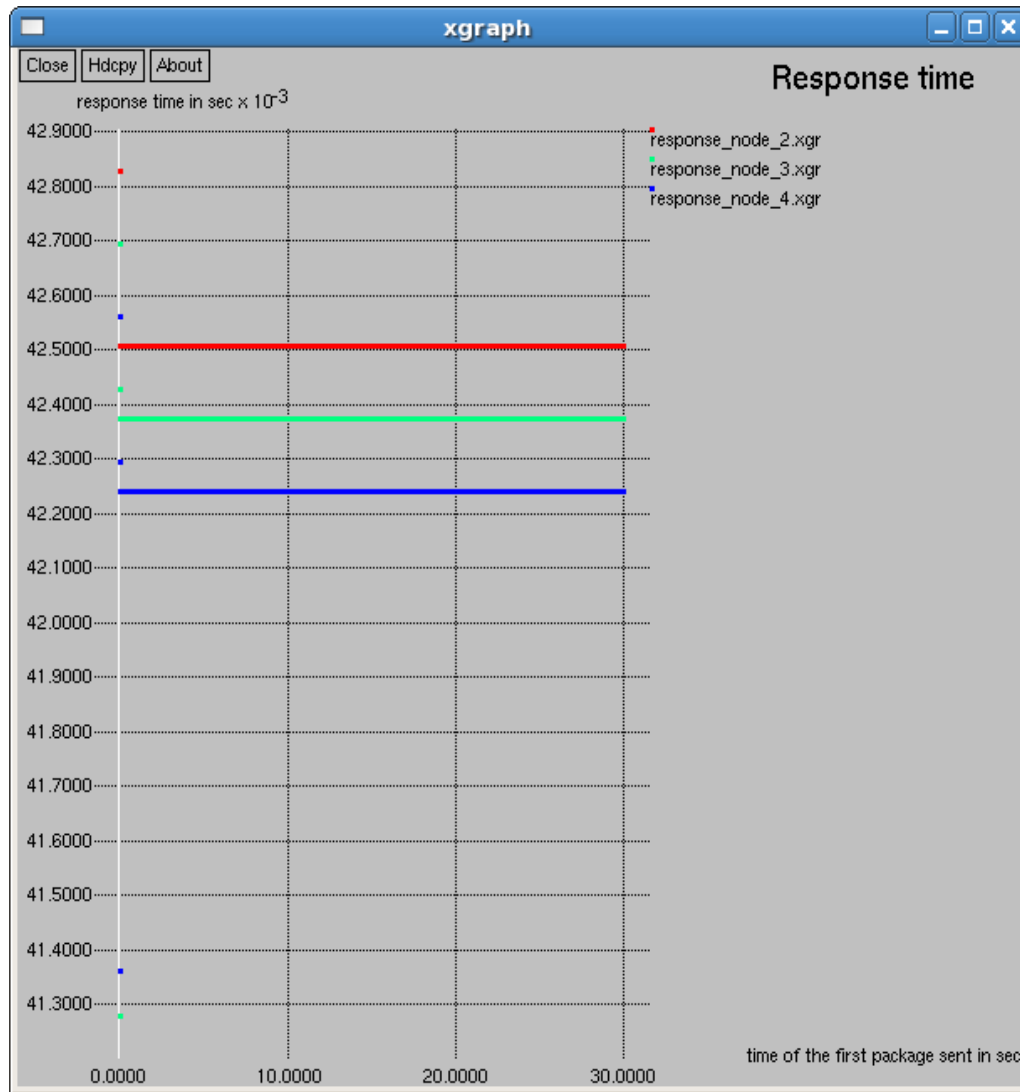


Abbildung 10: Antwortzeit ohne Restart

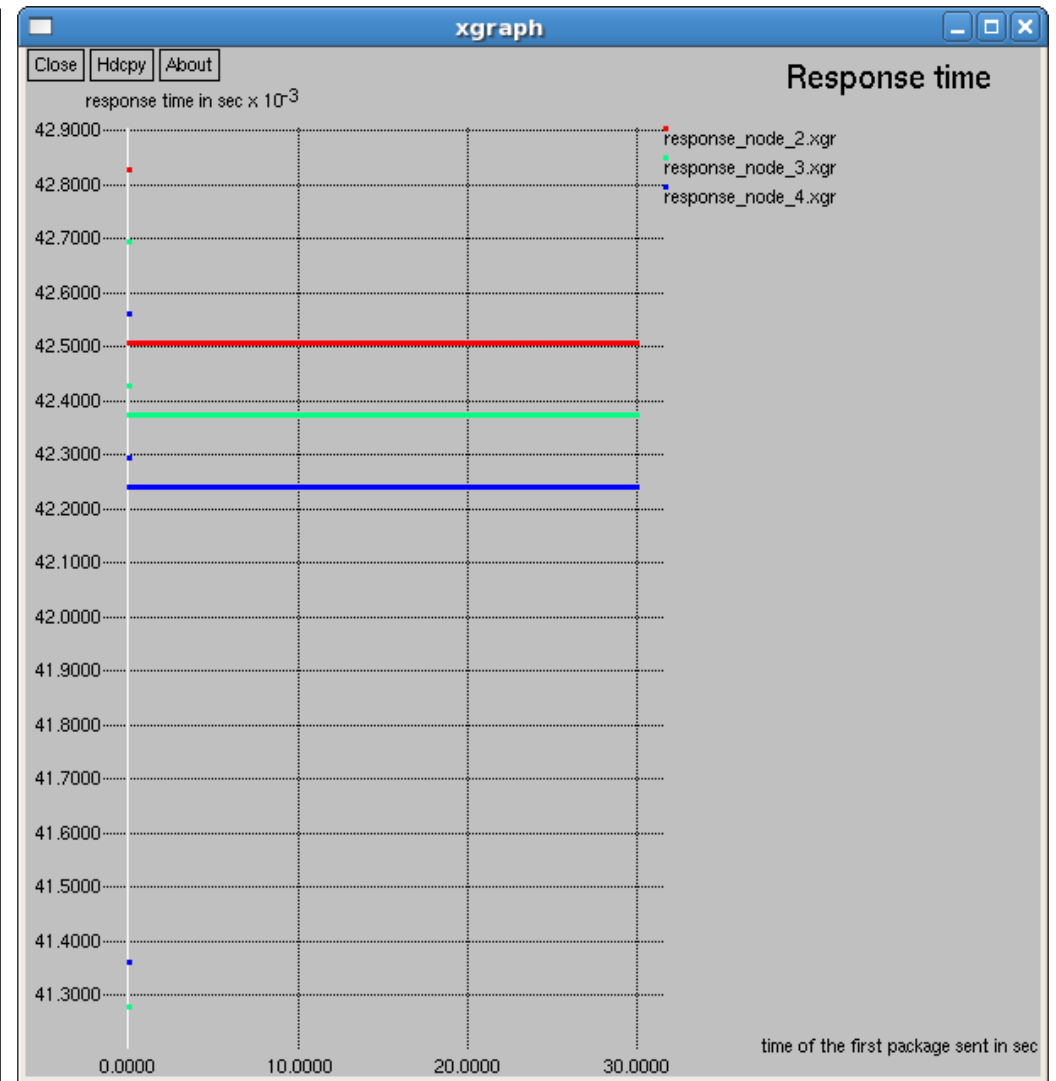


Abbildung 11: Antwortzeit mit Restart



Abbildung 12: Retransmission- Timer mit Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of packages sent (without ack-pck)	33687	33687
Number of acknowledge packages sent:	33663	33663
Number of packages dropped (without ack-pck)	0	0

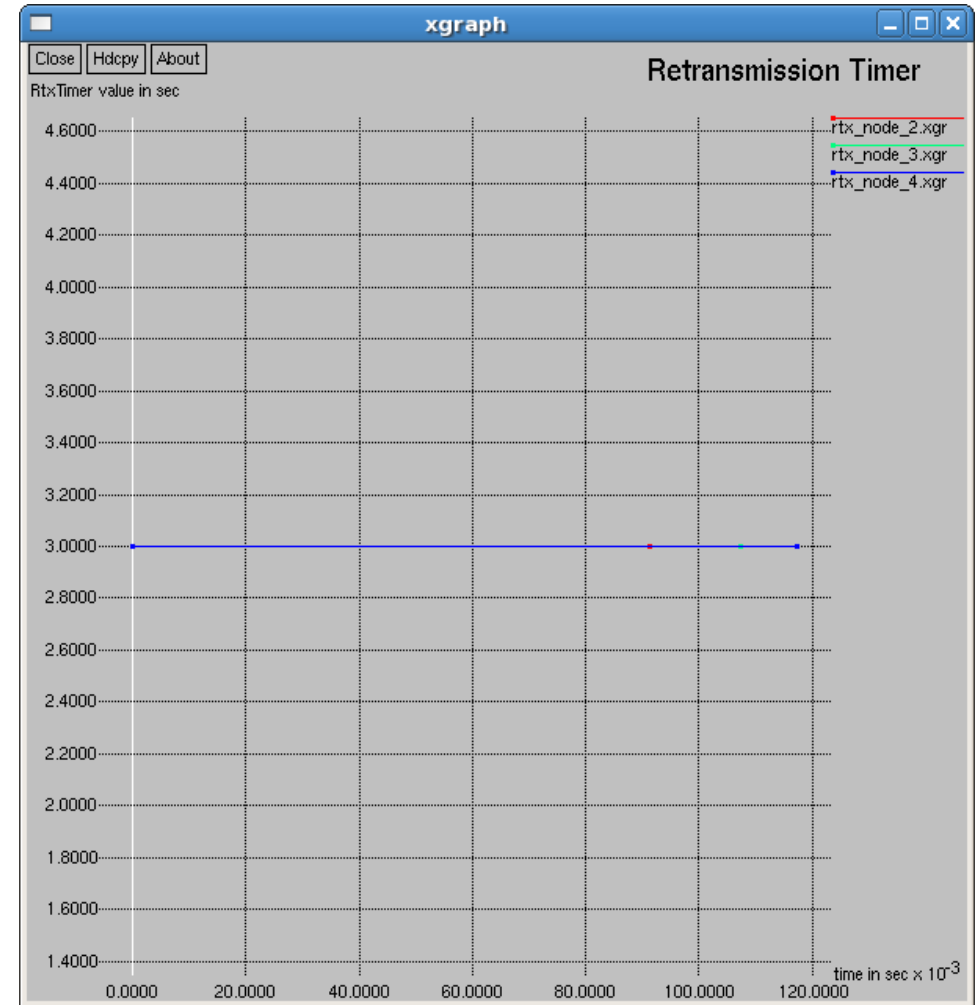


Abbildung 13: Retransmission- Timer ohne Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of acknowledge packages dropped	0	0
Number of acknowledged packages	33639	33639

In den beiden zuvor grafisch dargestellten Simulationen beträgt die Übertragungsrates der Links von Knoten 2,3,4 zu Knoten 1 jeweils 0,3 Mb. Auf dem Link 1-0 ergibt sich somit eine Last von 0,9 Mb die unterhalb der Leistungsgrenze dieses Links liegt. Die Ergebnisse bestätigen die Erwartungen, dass sich das Netz sowohl mit Restart, als auch ohne Restart gleich verhält, da keine erneute Übertragung von Paketen notwendig ist. Die Antwortzeit für alle Pakete ist konstant sowohl in der Restartsimulation als auch im Vergleichsversuch. Der Unterschied bezüglich des Retransmission- Timers ist in der Berechnung des Timers begründet. Bei der Verwendung des Restartverfahrens wird die Antwortzeit der ankommenden Pakete zur Berechnung verwendet. Die Antwortzeit unterhalb der Leistungsgrenze ist minimal, daher wird der Retransmission-Timer reduziert.

Für große Pakete (1000 Byte pro Paket) weist der Retransmission- Timer keine Veränderung zu kleinen Paketen (100 Byte pro Paket) auf. Für beide Paketgrößen liegt der Wert des RTX- Timers mit Restart bei ca. 0,2 Sek., ohne Restart hat er den Default- Wert des TCP- Protokolls (3 Sek.). Die Antwortzeit für kleine Pakete beträgt um die 0,42 Sek. Dies ist in etwa 0,15 Sekunden schneller als bei großen Paketen. Die geringeren Pakete wandern schneller über den Link, da sie nicht so viel Zeit zur Übertragung benötigen.

### 3.6 Übertragungsrate 0,4 Mb pro Link

#### 3.6.1 Übertragungsrate 0,4 Mb und Paketgröße 1000Byte

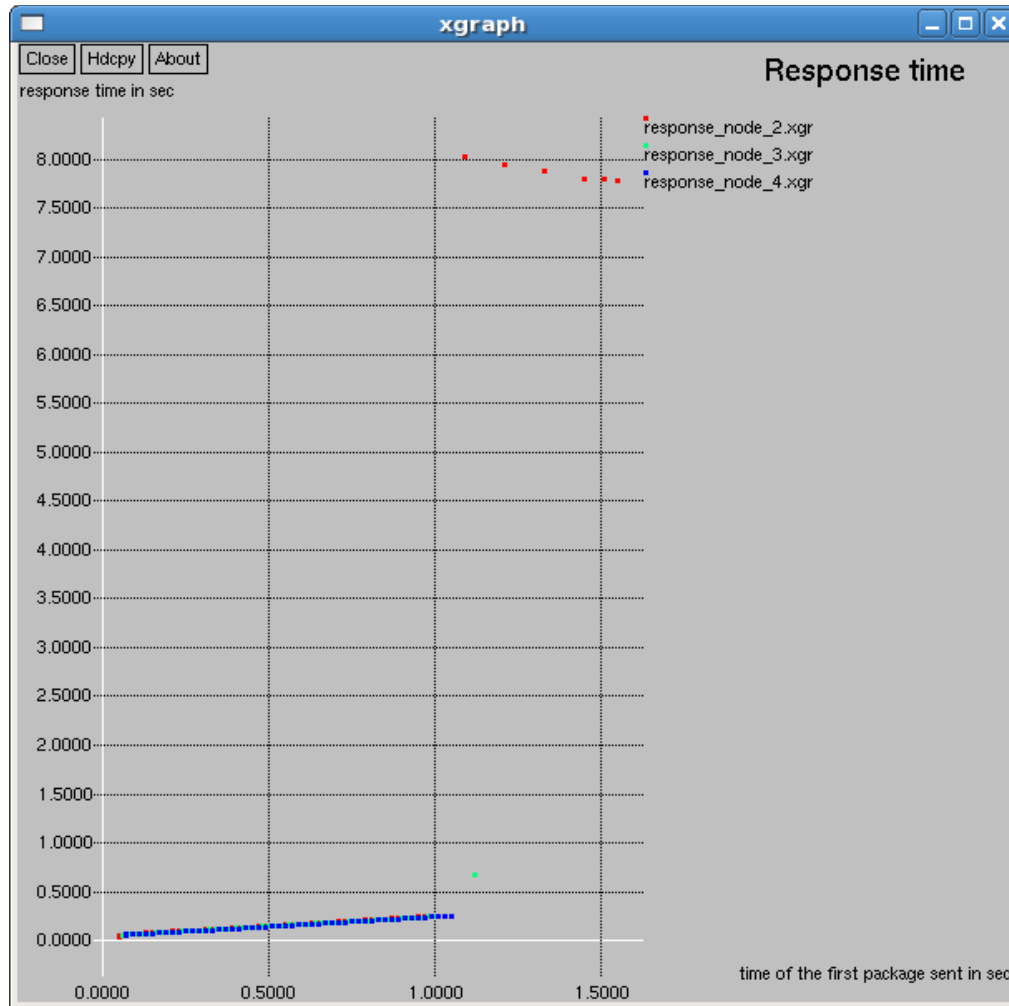


Abbildung 14: Antwortzeit mit Restart

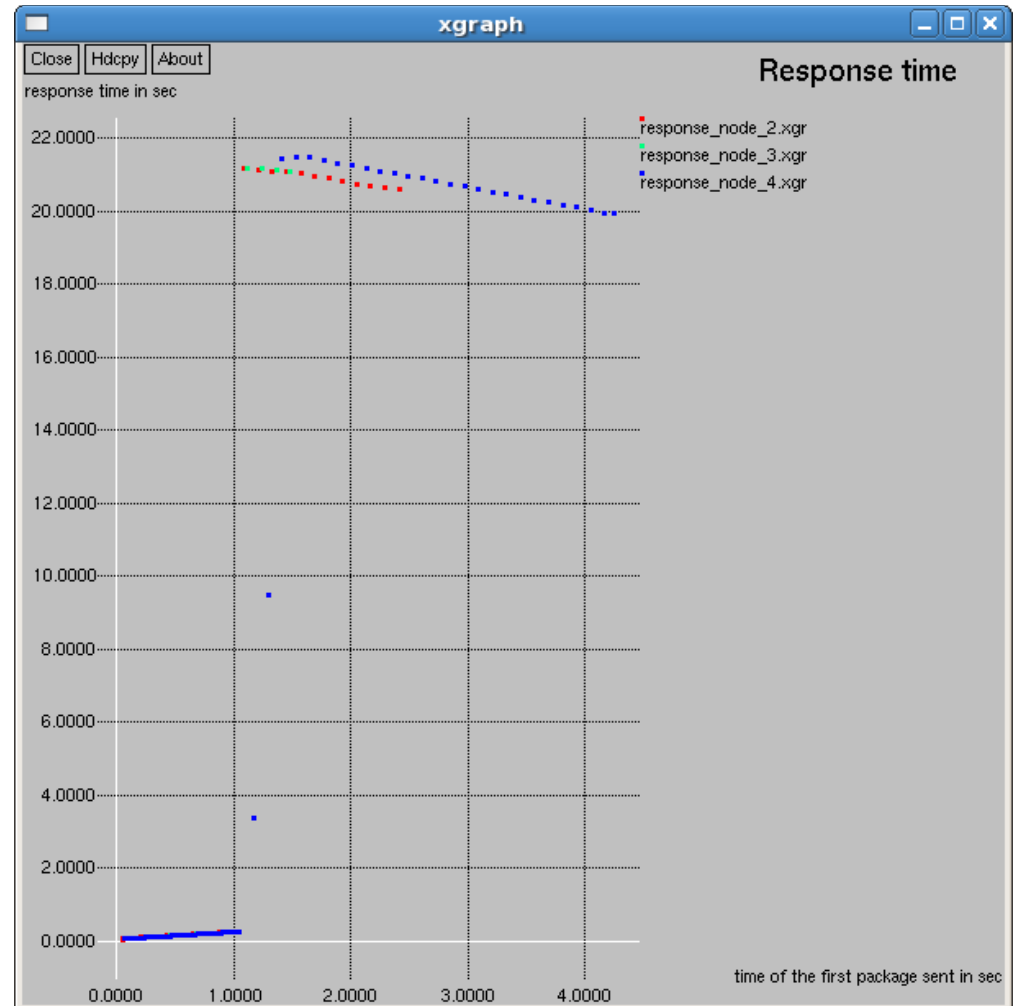


Abbildung 15: Antwortzeit ohne Restart

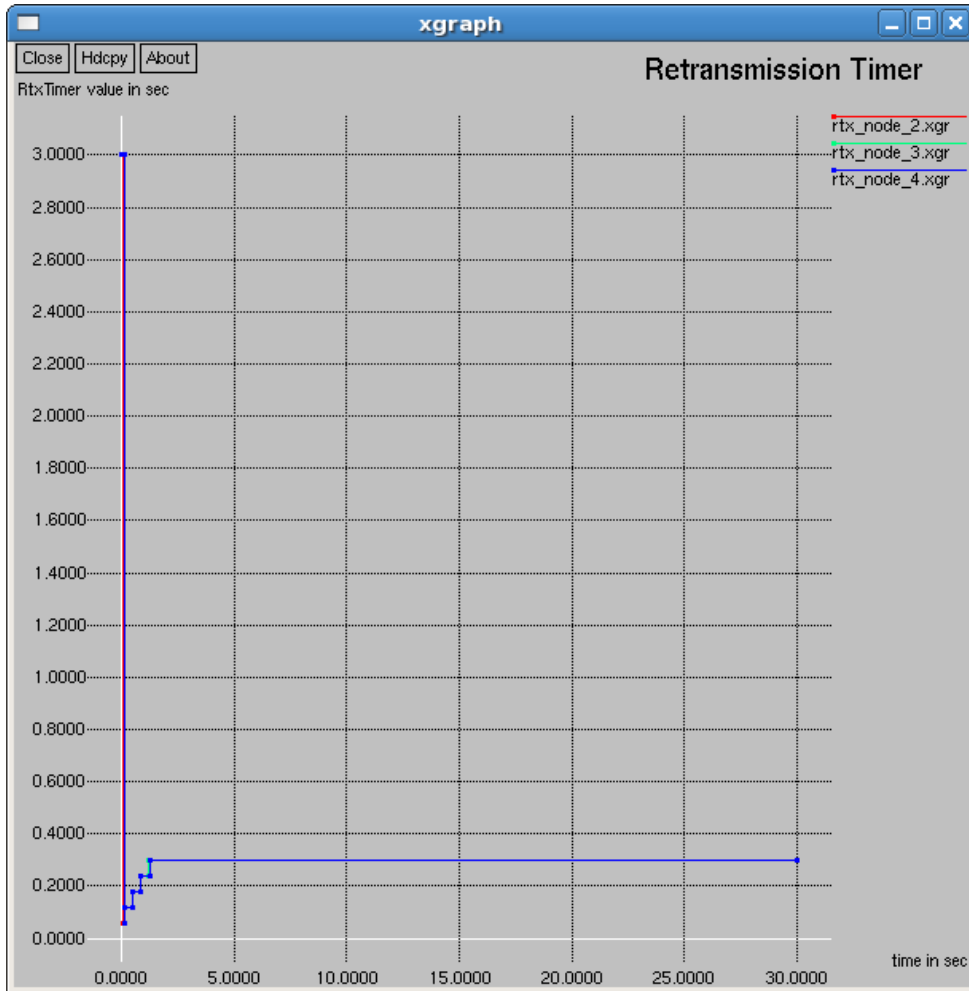


Abbildung 16: Retransmission- Timer mit Restart

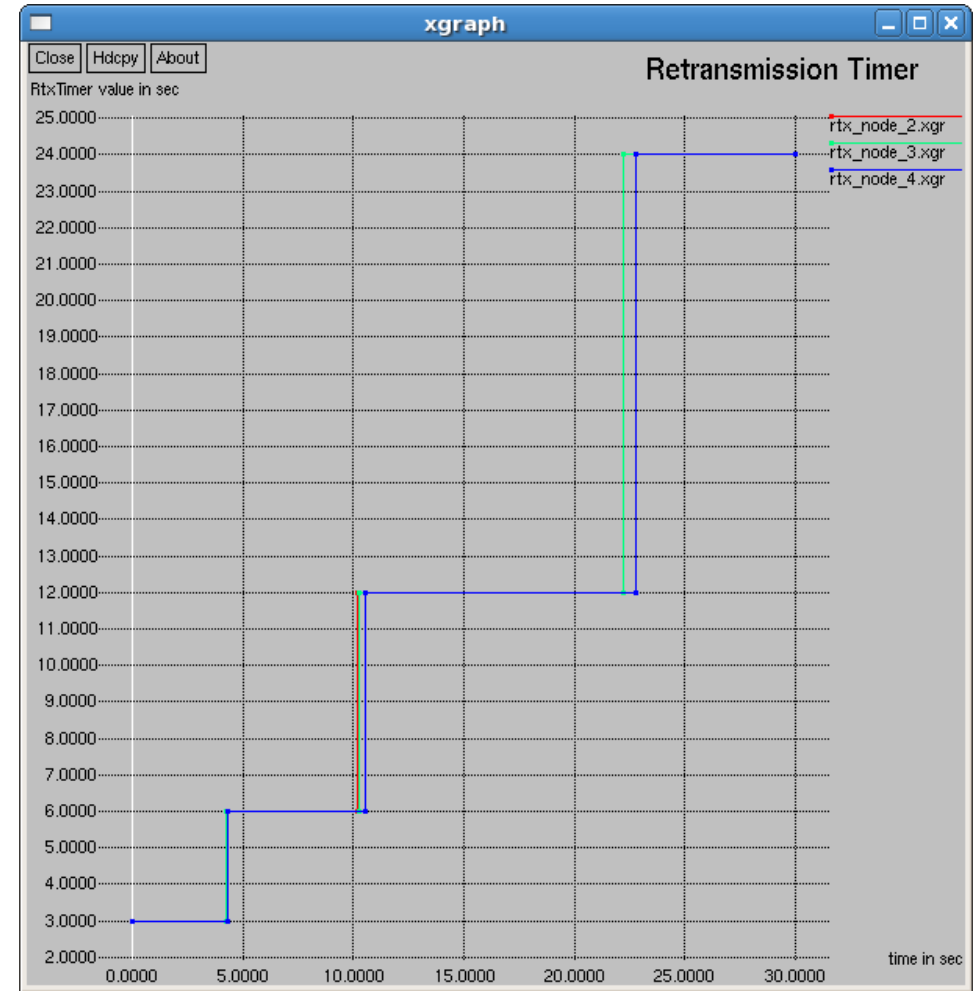


Abbildung 17: Retransmission- Timer ohne Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of packages sent (without ack-pck)	10882	7885
Number of acknowledge packages sent:	3742	3629
Number of packages dropped (without ack-pck)	7107	4222

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of acknowledge packages dropped	0	0
Number of acknowledged packages	155	191

Wie aus den Grafiken ersichtlich ist die Übertragungszeit nach dem Einschwingen des Systems deutlich geringer. Bis zur Simulationszeit von ungefähr 1 Sekunden verhalten sich beide Netze gleich. Die Antwortzeit steigt langsam an, bis auf dem Link 1-0 die Überlastsituation eintritt. Ab diesem Moment liegt die Antwortzeit im Netz mit Restart um 8 Sekunden, dies ist deutlich geringer als die 20-22 Sekunden des Netzes ohne Restart. Dieser Vorteil müsste sich auch in der Anzahl der bestätigten Pakete widerspiegeln. Hier tritt jedoch ein Widerspruch auf, da im Netz mit Restart die Anzahl der bestätigten Pakete 155 beträgt, während die Anzahl ohne Restart um 36 Pakete höher liegt. Addiert man die Zeit zu der das Paket gesendet wurde (ablesbar an der X-Achse) mit der Antwortzeit für dieses Paket erhält man die Zeit, zu der das Paket bestätigt wurde. Beim Restarternetz wurden bis zu einer Simulationszeit von 9,5 Sek. Pakete bestätigt danach nicht mehr. Durch die erhöhte Anzahl von Paketen auf dem Link wäre es möglich, dass die Bestätigungen auf dem Link 1-0 verworfen werden und dadurch die Bestätigungen ausbleiben. Diese Erklärung trifft jedoch nicht zu, da die Anzahl verlorengangener Bestätigungspakete null ist. Es muss eine andere Ursache geben: Da mehr Pakete erzeugt werden, aber weniger Pakete bestätigt werden, müssen die „selben“ Pakete (Pakete mit gleicher ID) öfter gesendet werden. Auf Grund dieser überhöhten Last werden mehr Pakete generiert, dadurch werden wiederum mehr Pakete verworfen. Eine Spirale entsteht, welche zu den beobachteten Ergebnissen führt.

Der Retransmission- Timer hat bereits bei 1,2 Sek seinen endgültigen Wert erreicht. Er ändert sich auch nicht nachdem keine Pakete mehr bestätigt werden. Die fehlende Anpassung des Timers ist letztendlich auch die Ursache für den erhöhten Verkehr. Abschließend kann also festgehalten werden, dass die fehlende Anpassung des Restart- Timers zu einer Verschlechterung des Ergebnisses führt.

### 3.6.2 Übertragungsrate 0,4 Mb und Packetgröße 100Byte

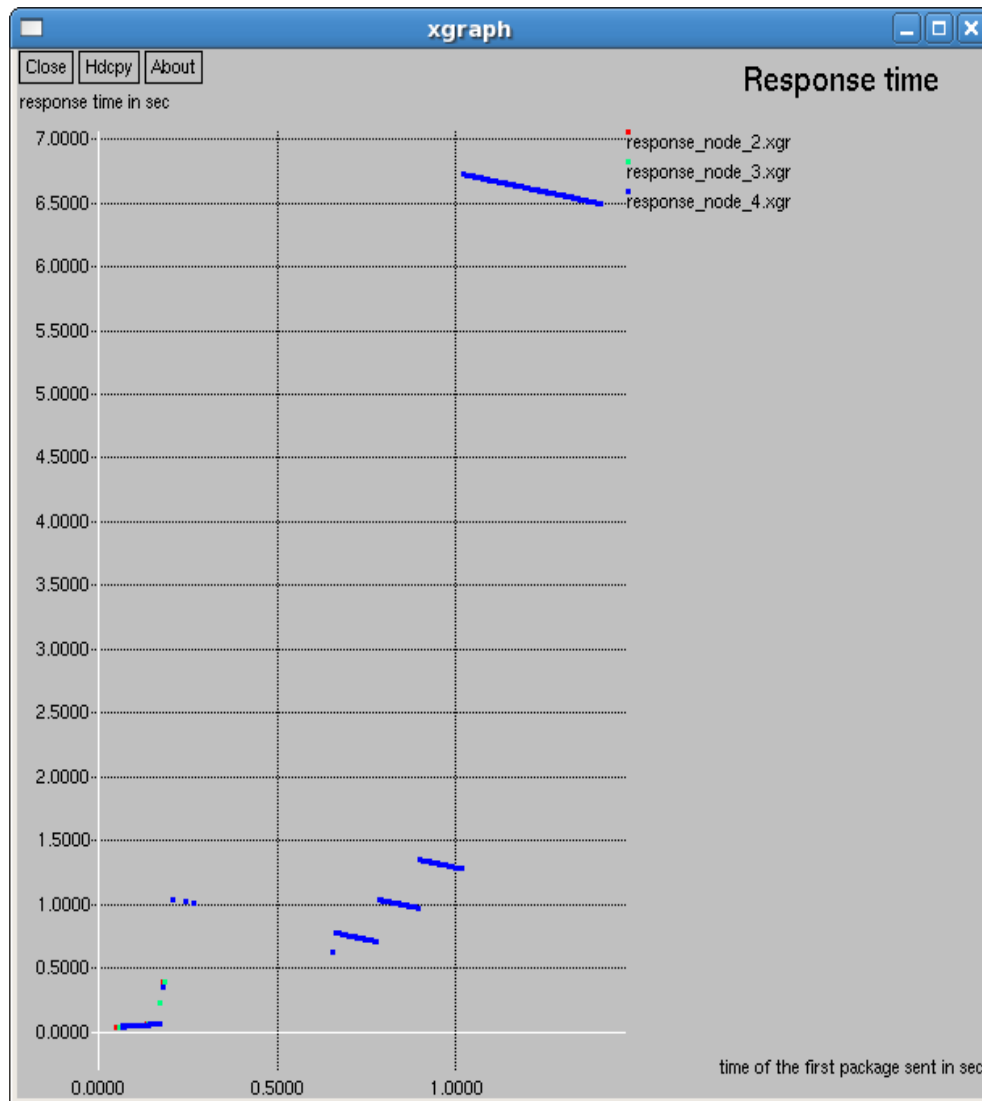


Abbildung 18: Antwortzeit mit Restart

Simulation ohne Restart nur bis zur Simulationszeit von 7,9 Sekunden

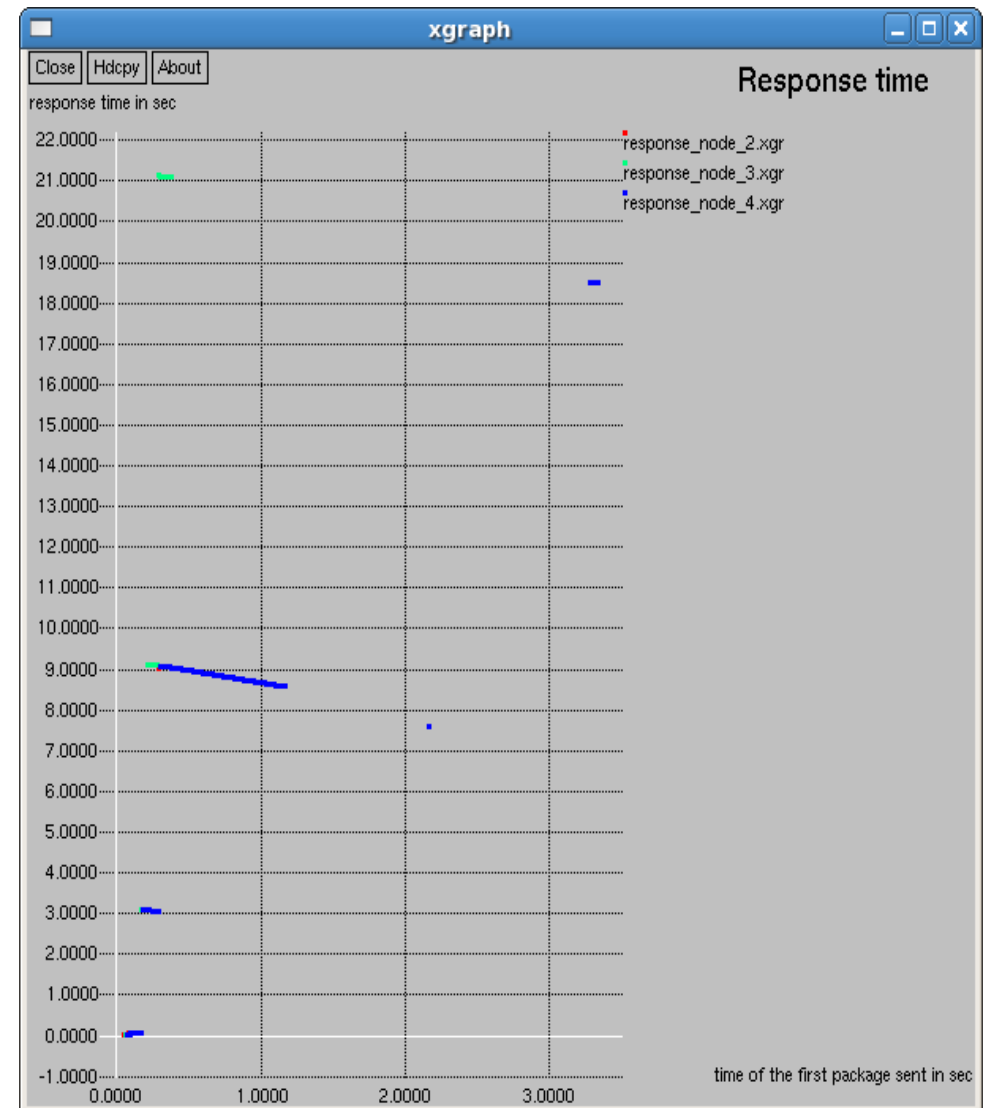


Abbildung 19: Antwortzeit ohne Restart

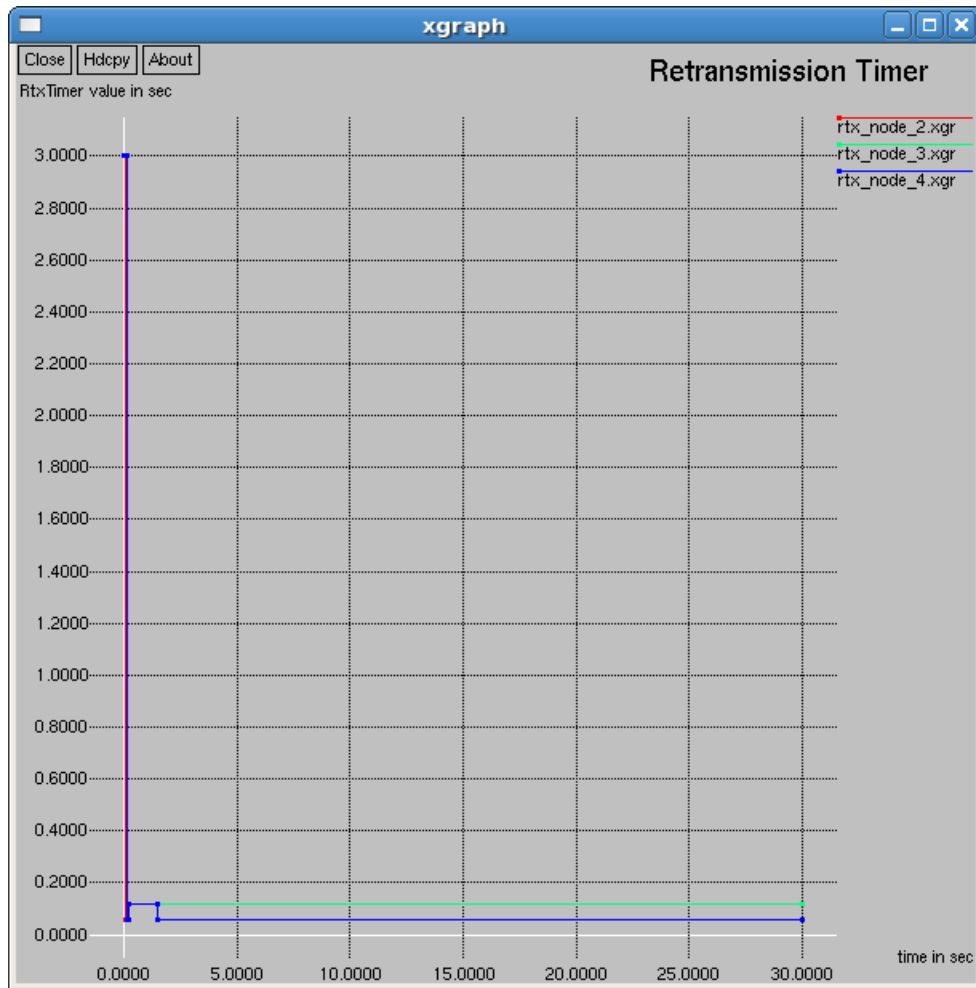


Abbildung 20: Retransmission- Timer mit Restart

Simulation ohne Restart nur bis zur Simulationszeit von 7,9 Sekunden

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of packages sent (without ack-pck)	28697	13434
Number of acknowledge packages sent:	9786	7295
Number of packages dropped (without ack-pck)	18832	6139

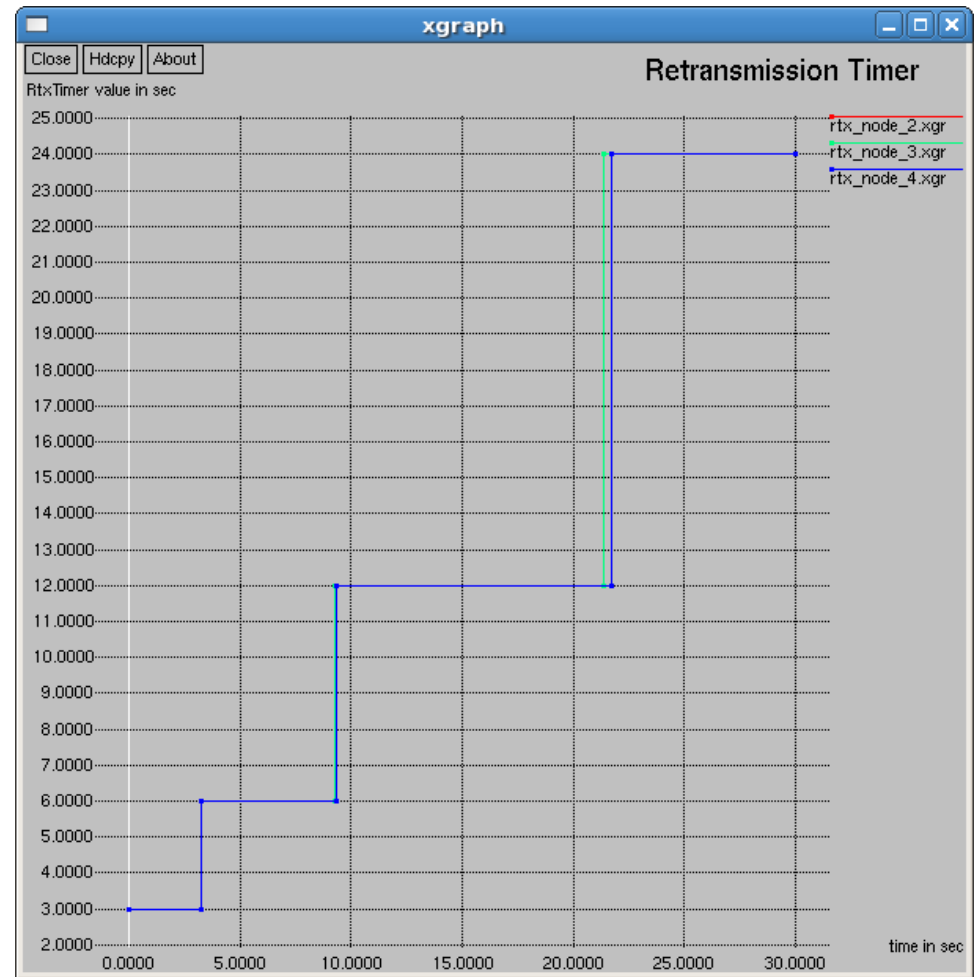


Abbildung 21: Retransmission- Timer ohne Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of acknowledge packages dropped	0	0
Number of acknowledged packages	297	283

Der Vergleich zwischen Restart und nicht Restart bei Netzen mit kleineren Paketen kann nicht vollständig geführt werden, da auf Grund von Beschränkungen in der Simulationshardware das Restartnetz nur bis zu einer Zeit von 8 Sekunden simuliert werden konnte. Es zeigt sich, dass auch bei dieser Simulation eine erhebliche Anzahl von verlorengegangenen Paketen auftritt. In Gegensatz zu den Simulationen mit großen Paketen (siehe: 3.6.1) übersteigt die Anzahl von bestätigten Pakten im Restartnetz die Paketmenge im Netz ohne Restart bereits dann, wenn das Restartnetz kürzer simuliert wird. Daher läßt sich vermuten, dass der Restart- Algorithmus bei kleinen Paketen effektiver ist. Eine weitere Möglichkeit wäre, dass die Protokollparameter (wie zum Beispiel das Übertragungsfenster) optimaler für kleine Pakete sind.

Der Einfluß der Paketgröße ist bei gleichen Konfigurationen (d.h. mit bzw. ohne Restart) unerheblich. Beim Restartnetz mit kleinen Paketen liegt die Antwortzeit im gleichen Bereich wie bei dem selben Netz mit großen Paketen. Die gleiche Beobachtung trifft auch für Netze ohne Restart zu. Die Retransmission- Timer ermöglichen analoge Beobachtungen zu den Antwortzeiten bei der Gegenüberstellung von großen zu kleinen Paketen.

### 3.7 Übertragungsrate 1,0 Mb pro Link

#### 3.7.1 Übertragungsrate 1,0 Mb und Paketgröße 1000Byte

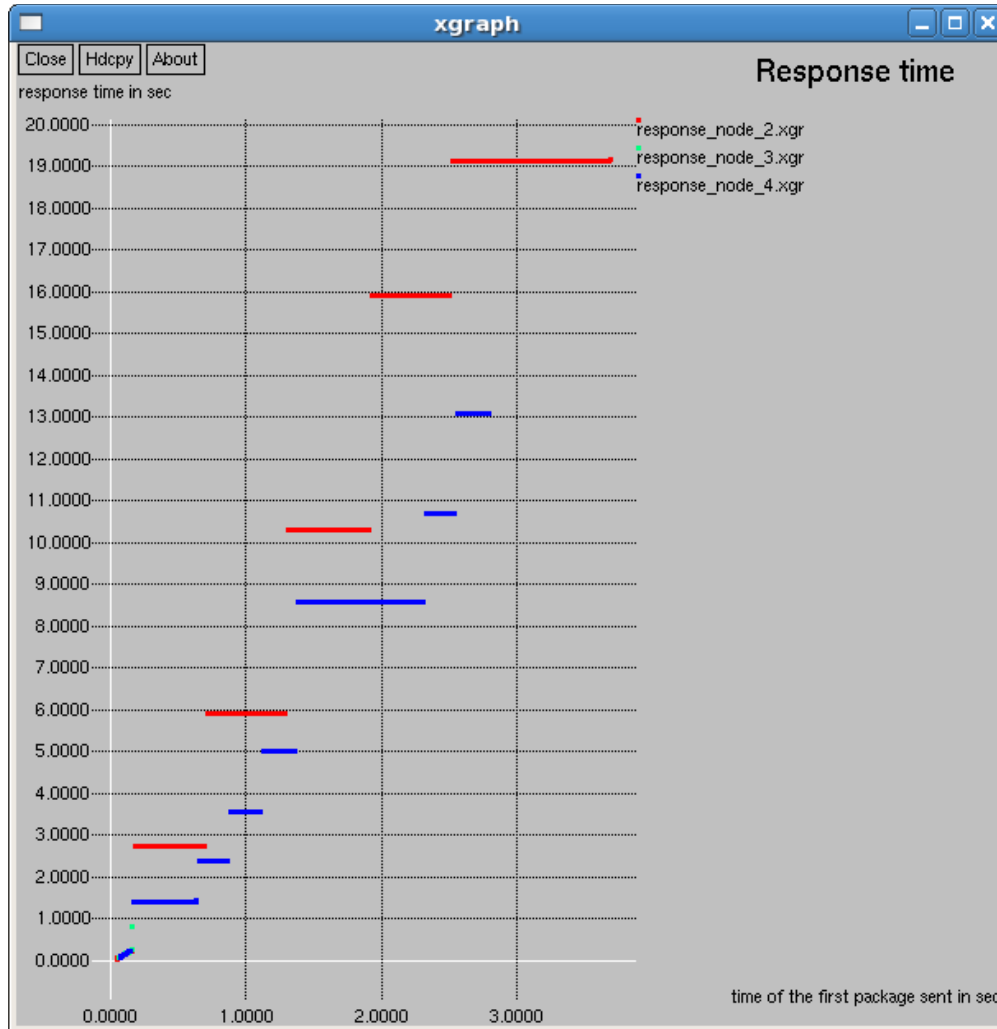


Abbildung 22: Antwortzeit mit Restart

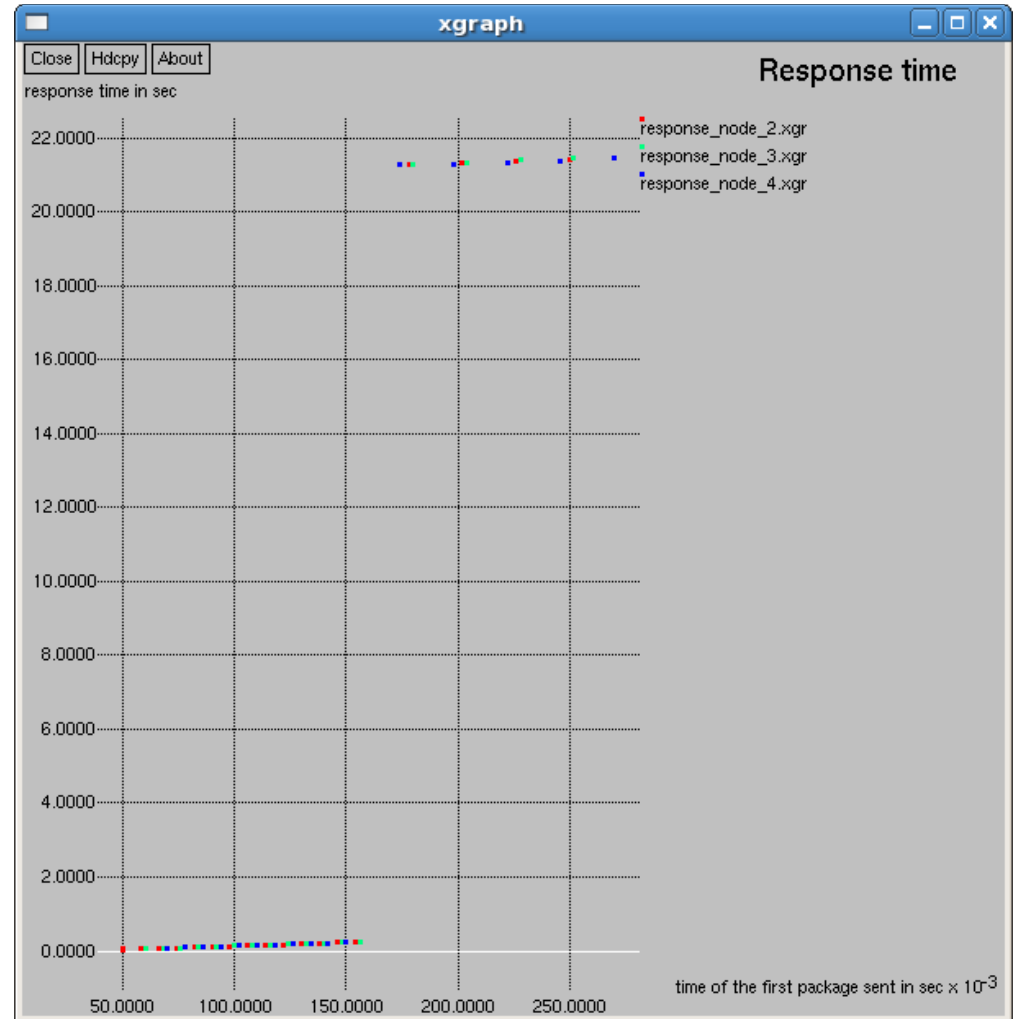


Abbildung 23: Antwortzeit ohne Restart

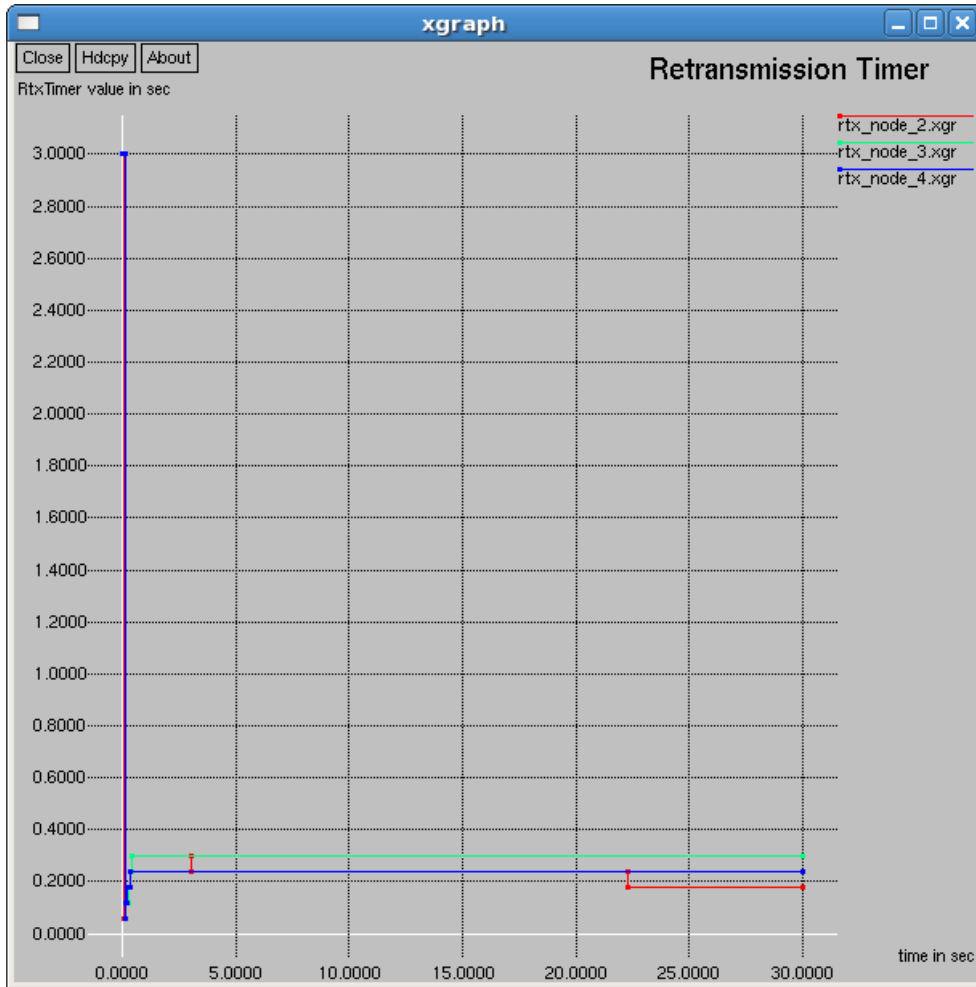


Abbildung 24: Retransmission- Timer mit Restart

Simulation ohne Restart nur bis zur Simulationszeit von 22,9 Sekunden

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of packages sent (without ack-pck)	8613	10294
Number of acknowledge packages sent:	2870	3482
Number of packages dropped (without ack-pck)	5711	6812

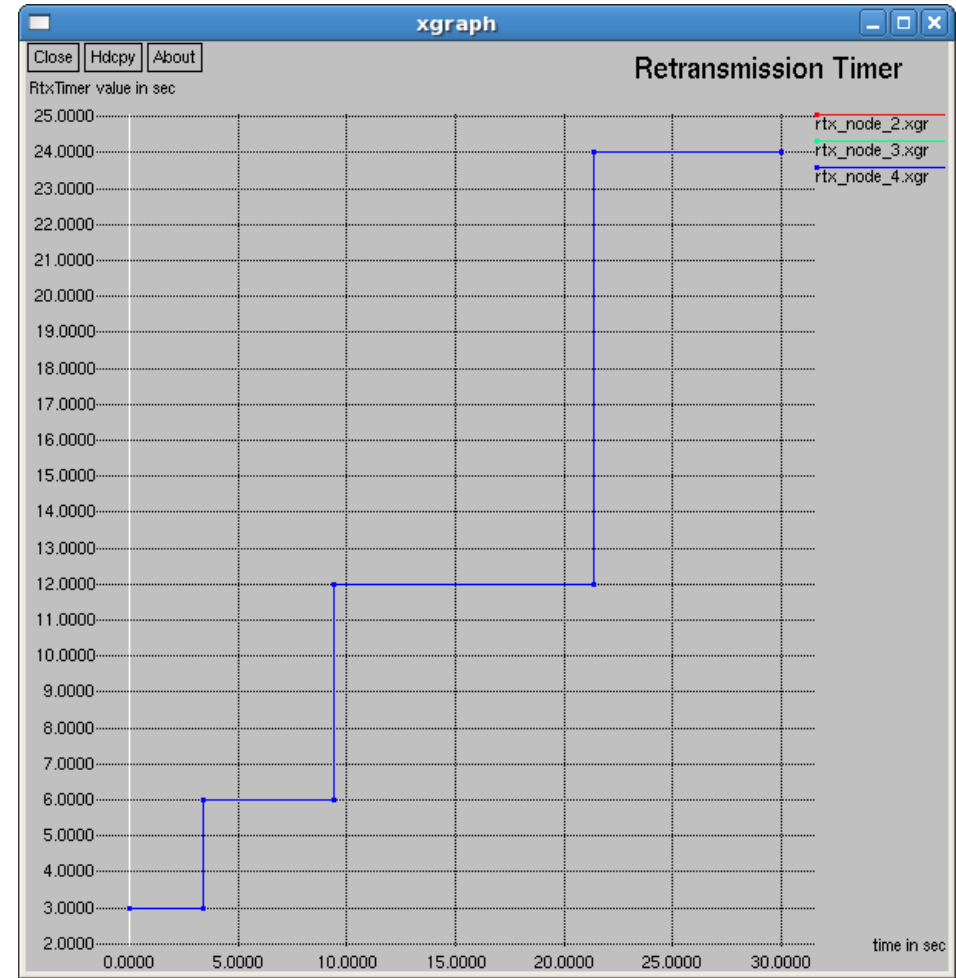


Abbildung 25: Retransmission- Timer ohne Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of acknowledge packages dropped	0	0
Number of acknowledged packages	305	54

### 3.7.2 Übertragungsrate 1,0 Mb und Packetgröße 100Byte

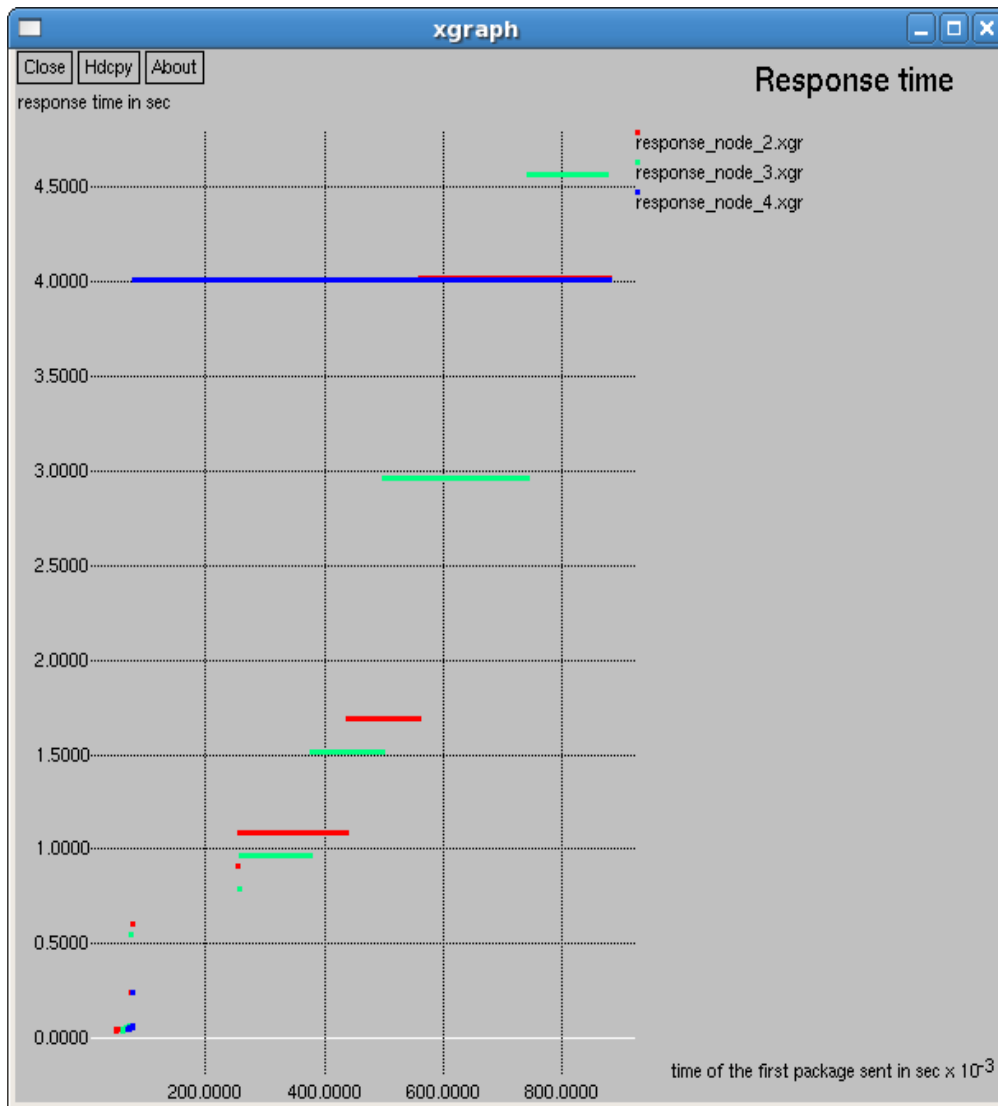


Abbildung 26: Antwortzeit ohne Restart

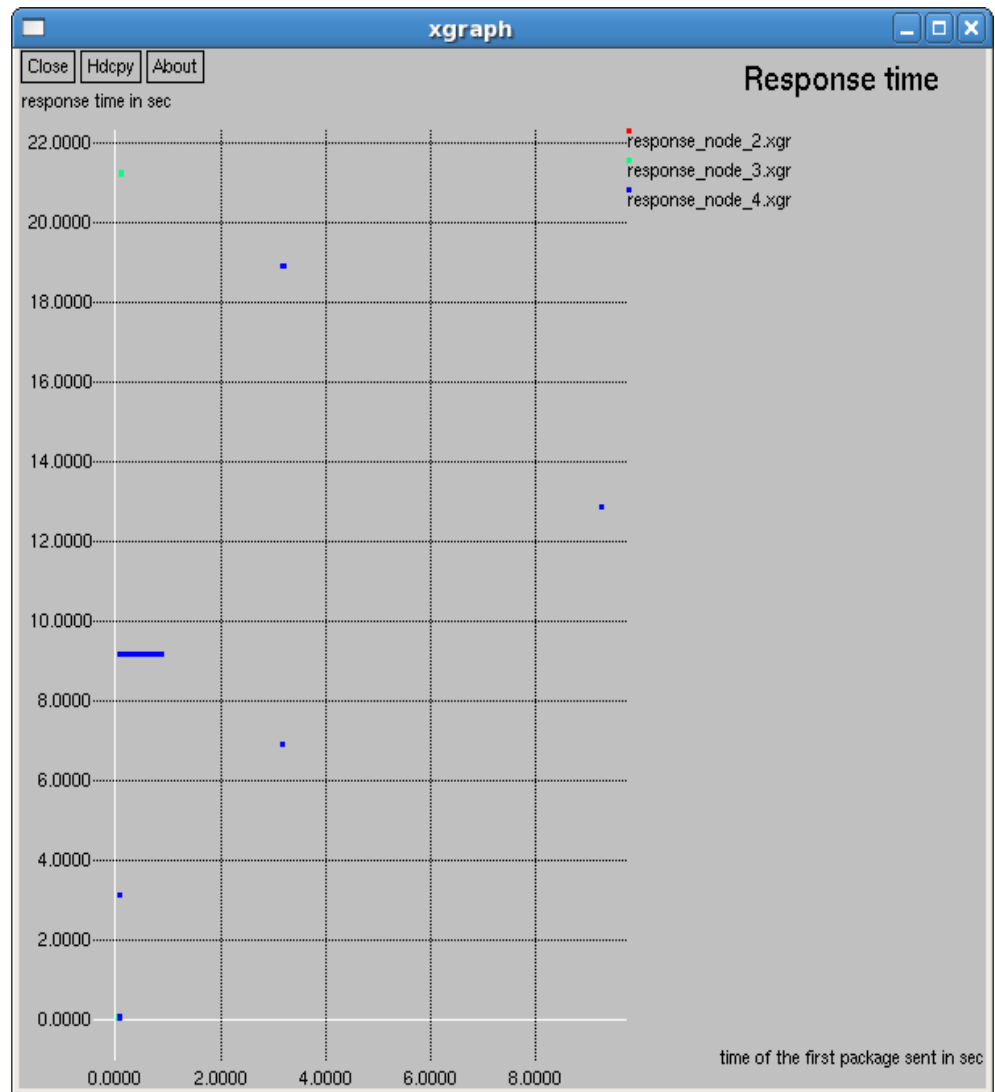


Abbildung 27: Antwortzeit ohne Restart

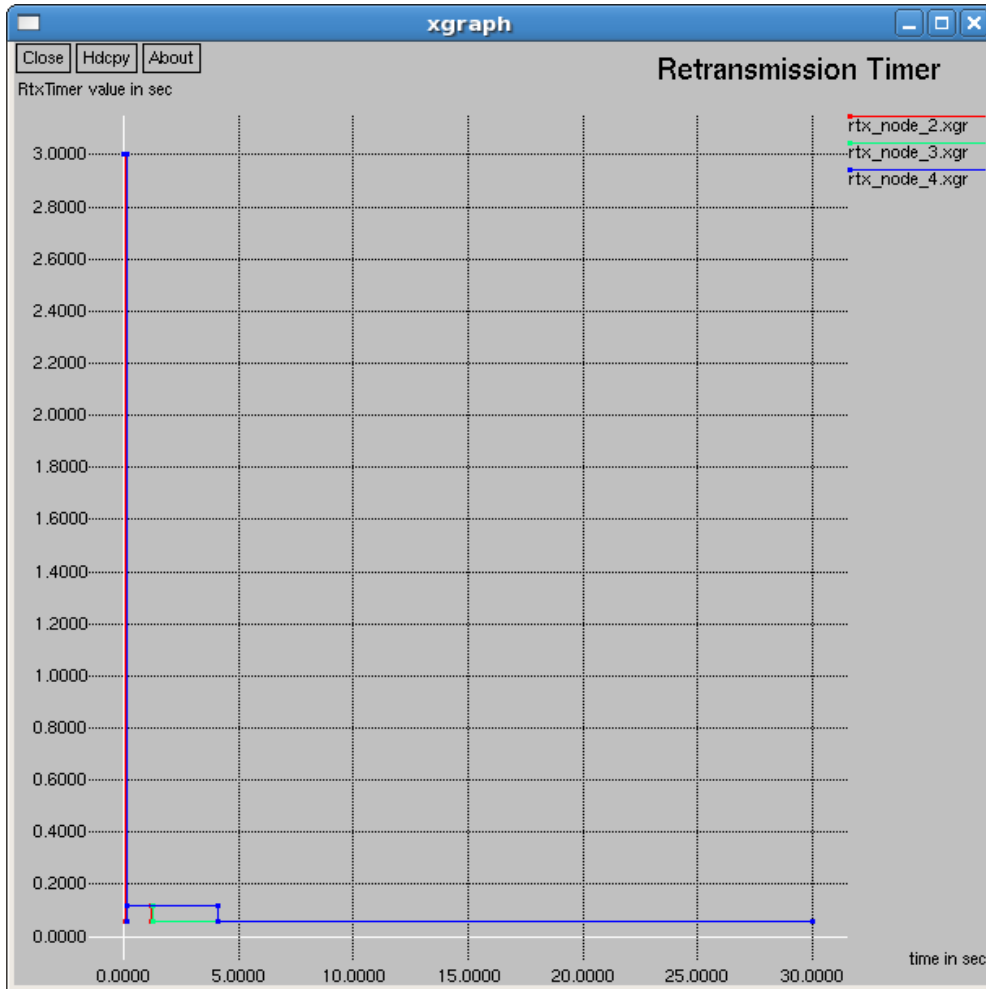


Abbildung 28: Retransmission- Timer mit Restart

Simulation ohne Restart nur bis zur Simulationszeit von 6,0 Sekunden

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of packages sent (without ack-pck)	8613	10294
Number of acknowledge packages sent:	2870	3482
Number of packages dropped (without ack-pck)	5711	6812

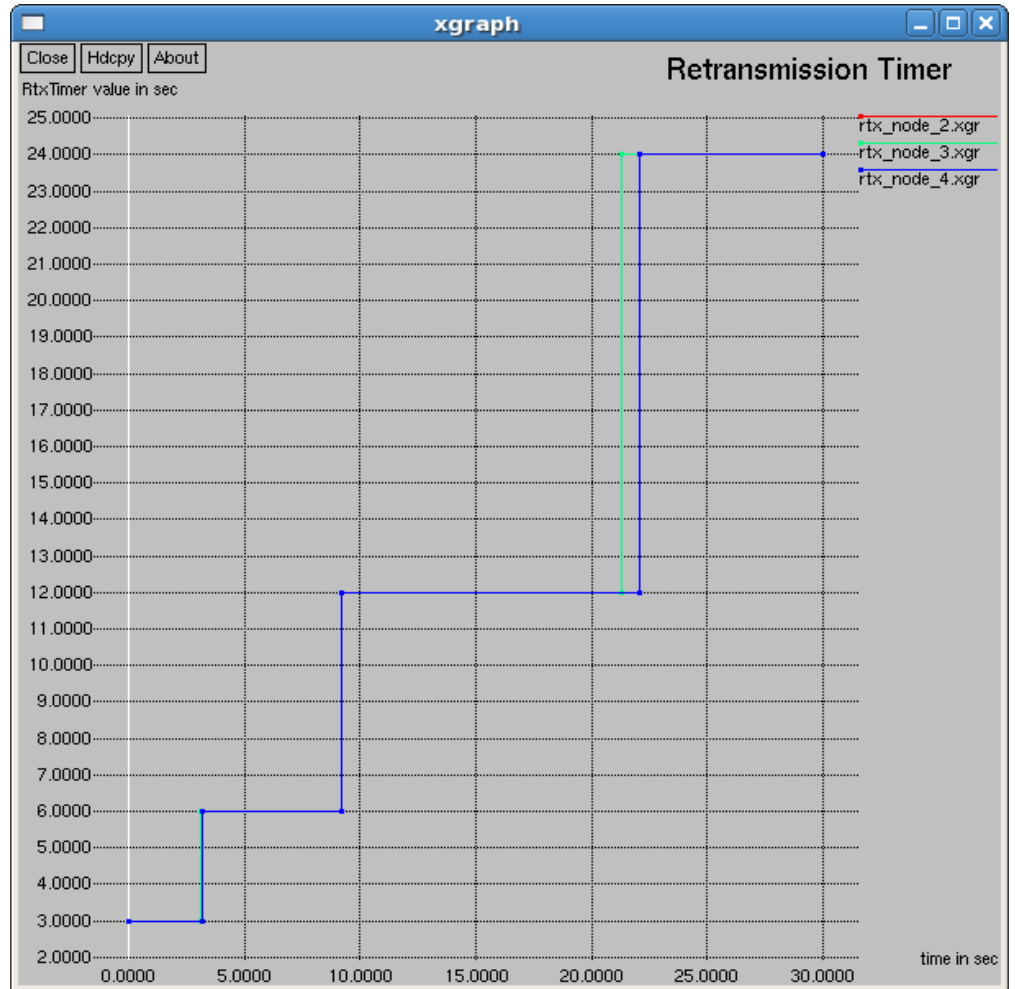


Abbildung 29: Retransmission- Timer ohne Restart

	<b>Mit Restart</b>	<b>Ohne Restart</b>
Number of acknowledge packages dropped	0	0
Number of acknowledged packages	305	54

Die Untersuchung der Ergebnisse für die höchste Last ist nur unter Vorbehalt möglich, da sowohl für kleine Pakete als auch große Pakete die Simulation mit Restart nur bis zu 6 Sekunden Simulationszeit aufgrund von unzureichender Simulationshardware möglich war.

Die Ergebnisse der Analyse Restart vs nicht Restart sind analog zu den Aussagen aus 3.6.1. Im vorliegenden Fall erzeugt der Restartalgorithmus in der selben Simulationszeit mehr Pakete, wenn man davon ausgeht, dass die Anzahl der Pakete sich kontinuierlich wie im Intervall 0 .. 6 Sekunden entwickelt. Gleichzeitig steigt die Anzahl der bestätigten Pakete bereits in den ersten Sekunden über die Anzahl von Paketen in Netzen ohne Restart. Der Restartalgorithmus bewirkt eine höhere Anzahl von bestätigten Paketen.

Bei dieser Simulation zeigt sich besonders, dass die Paketgröße keinen Einfluß auf die Simulation hat. Die Zahlen über die Paketmengen stellen dies sehr deutlich dar. Sowohl für große Pakete als auch kleine Pakete werden in den entsprechenden Simulationen die selbe Anzahl von Paketen erzeugt, verworfen und bestätigt.

### ***3.8 Zusammenfassung:***

Die Größe der Pakete hat bei steigender Auslastung weniger Einfluß auf die Anzahl der bestätigten Pakete. Innerhalb einer Simulationsreihe (0,3Mb – 0,4Mb – 1,0Mb pro Route) mit gleicher Auslastung ist der Einfluß der Paketgröße auf die einzel Werte der Antwortzeiten und des Retransmission- Timer vernachlässigbar. Sowohl bei großen als auch bei kleinen Paketen werden für die einzelnen Pakete ähnliche Werte erzielt.

In der Regel werden in einem Netz mit Restart mehr Pakete bestätigt. Dabei muss auch berücksichtigt werden, dass die Anzahl gesendeter Pakete erheblich höher ist, was zu einer stärkeren Belastung des Netzes führt. Aus Sicht der Anwendung ist dies uninteressant, da mehr Daten bestätigt und somit übertragen werden.

Verwunderlich ist, dass dieses Verhalten nicht auftrat bei einer Übertragungsrate von 0,4 Mb und großen Paketen (vgl: 3.6.1). In der genannten Simulation werden im Restart- Netz weniger Pakete bestätigt als im Vergleichsnetz. Es lässt sich vermuten, dass die Netzwerkparameter nicht optimal für diese Konfiguration mit Restart sind.

## Bibliographie

1. Andreas Mosig:  
**Net Simulator 2, Routing Seminar**  
Universität Koblenz-Landau, 2004  
<http://www.uni-koblenz.de/~steigner/seminar-routingsim/mosig.pdf> (Stand: 18.02.2007)
2. Chrisian Hofmann, Christoph Lumme:  
**Praktikum Netzwerksimulator 2, Tutorial**  
Im Rahmen des Projektes SIMON (Simulation Environment for Mobile Networks[3]), Technische Universität Ilmenau, 2005  
[http://www.tu-ilmenau.de/fakia/fileadmin/template/startIA/ihss/dat/lehre/fak\\_prak/tutorial\\_ns2praktikum.pdf](http://www.tu-ilmenau.de/fakia/fileadmin/template/startIA/ihss/dat/lehre/fak_prak/tutorial_ns2praktikum.pdf) (Stand: 18.02.2007)
3. Eitan Altman, Tania Jiménez:  
**NS Simulator for beginners, Lecture notes**  
Univ. de Los Andes, Mérida, Venezuela and ESSI, 2003-2004  
<http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/COURS-NS/n3.pdf> (Stand: 18.02.2007)
4. Katinka Wolter, Aad P. A. van Moorsel:  
**Analysis and Algorithms for Restart**  
Humboldt-Universität zu Berlin, Institut für Informatik; University of Newcastle, School of Computing Science  
<http://www2.informatik.hu-berlin.de/~wolter/publications/quest04.ps> (Stand: 20.02.2007)
5. Katinka Wolter, Philipp Reinecke, Aad van Moorsel:  
**A Measurement Study of the Interplay Between Application Level Restart and Transport, Protocol**  
Humboldt-Universität zu Berlin, Institut für Informatik  
<http://www2.informatik.hu-berlin.de/~wolter/publications/isas04.pdf> (Stand: 20.02.2007)
6. Kevin Fall, Kannan Varadhan:  
**The ns Manual, The VINT Project**  
A collaboration between researchers at UC Berkley, LBL, USC/ISI, and Xerox PARC, 2007  
<http://www.isi.edu/nsnam/ns/ns-documentation.html> (Stand: 18.02.2007)
7. **Ns2-Hauptseite**  
[http://nsnam.isi.edu/nsnam/index.php/Main\\_Page](http://nsnam.isi.edu/nsnam/index.php/Main_Page) (Stand: 18.02.2007)
8. **Seite von VINT-Projekt**  
<http://www.isi.edu/nsnam/vint/> (Stand: 18.02.2007)
9. **Download-Seite von ns2-2.30**  
<http://mailman.isi.edu/pipermail/ns-users/2006-September/057343.html> (Stand: 20.02.2007)
10. **Seite von cygwin:** <http://www.cygwin.com/> (Stand: 20.02.2007)