Diplomarbeit

VoIP in Unternehmensnetzen

Simulation von QoS Mechanismen zur Überprüfung der Einsetzbarkeit in bestehenden IP Netzen mittels des NS-2 Netzwerk Simulators

Ausgeführt zum Zweck der Erlangung des akademischen Grades

Dipl.-Ing. (FH) für Telekommunikation und Medien am Fachhochschul-Diplomstudiengang Telekommunikation und Medien St. Pölten

unter der Leitung von

Dipl.-Ing. Johann Haag

Zweitbegutachtung

Dipl.-Ing. Georg Barta

ausgeführt von

Florian Sperker

Tm0110038108

Wien, 06.06.2006

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe,
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der vom Begutachter beurteilten Arbeit überein.

.....

Wien, 20.06.2006

Danksagung

Ich möchte mich bei folgenden Leuten für die Unterstützung während meiner Arbeit bedanken: Meinen Eltern für ihre mentale Unterstützung, meinen Freunden Florian Holzinger und Marius Oskandi sowie meinem Cousin Jakob, welche Teile ihrer Freizeit geopfert haben um mir bei den Programmierproblemen mit dem NS-2 Simulator zu helfen.

Zu besonderem Dank verpflichtet fühle ich mich Herrn Dipl.-Ing. Haag, der durch zahlreiche Anregungen und Verbesserungsvorschläge nicht unwesentlich zum Gelingen dieser Arbeit beitrug.

Kurzfassung

Mit der zunehmenden Entstehung immer größerer Netzwerke stellt sich vor allem in größeren Firmen immer häufiger die Frage, ob sich der Umstieg auf netzwerkbasierende Telefonsysteme, kurz VoIP (*Voice over IP*) rechnet. Allerdings sind die meist inhomogen gewachsenen Netzwerke nicht für solche Dienste ausgelegt worden und es ist fraglich, inwieweit sich ein solch komplexes und kritisches System wie VoIP in das eigene Netz integrieren lässt und welche Änderungen oder technische Zusätze dafür notwendig sind. Das Grundproblem liegt darin, dass Sprachübertragungen äußerst zeit- und bandbreitenkritisch sind, und das TCP/IP Netzwerksystem genau diese technische Vorausetzung nicht bietet. Die Netzwerktechnik bedient sich hierbei verschiedener QoS (*Quality of Service*) Mechanismen, um dieses Problem zu lösen.

In dieser Arbeit werden die verschiedenen Möglichkeiten von QoS analysiert und mittels des Berkeley NS-2 Netzwerksimulators getestet, um fundierte Aussagen über die Einsetzbarkeit von QoS Mechanismen für VoIP zu ermöglichen.

Abstract

Given the advent of ever larger networks, the issue particularly at large-scale operations increasingly arises whether a shift toward network-based telephony systems, VoIP (Voice over IP) for short, pays off. However, the networks, mostly heterogeneously "grown", have not been conceived for such services and it remains to be seen whether and to what extent an intricate and involved system such as VoIP can be integrated into the proprietary network and which modifications or system-related add-ons are required to that effect. The basic issue resides in the fact that voice transfer is extremely sensitive to both time and bandwidth (range), and it is this very requirement that the TCP/IP network system fails to meet. To that effect, the network technology draws upon QoS (Quality of Service) techniques in order to solve this problem.

The present paper analyzes the different applications of QoS, testing them by drawing upon the Berkeley NS-2 network simulator, in order to make meaningful and authoritative statements about the applicability of QoS techniques for the purpose of VoIP.

Inhaltsverzeichnis

Einleitung

Seit geraumer Zeit ist VoIP im Gespräch und in Verwendung. Früher noch wenig ausgereift und nur von Providern oder Computerfirmen genutzt, etabliert sich die Internettelefonie heutzutage immer stärker. Seit dem Boom der Breitbandinternetanschlüsse ist der Weg frei für VoIP Telefone für jeden Haushalt. Die technischen Anforderungen werden mittlerweile von allen gängigen DSL- oder Kabelanschlüssen erfüllt und deren Tarife sind ebenfalls günstig geworden. So sind VoIP Anlagen auch für Unternehmen interessant, da diese im Betrieb meist günstiger und flexibler sind als normale Telefonanlagen. Besonders die Anbindung von internationalen Firmenzweigstellen über VoIP ist vorteilhaft, wenn dadurch teure Auslandsgespräche vermieden werden können, da diese dann über das mittlerweile günstigere Internet anstatt die normalen Telefonleitungen geführt werden. Außerdem fällt die gesamte und sehr kostenintensive Telefonanlage samt Telefon-verkabelung weg.

Doch VoIP hat auch Nachteile. Die meisten großen Computernetze sind nicht für Echtzeitanwendungen wie VoIP gedacht. Das liegt daran, dass nahezu alle Netzwerke das Übertragungsprotokoll TCP/IP nutzen, welches nicht optimal für diese Aufgabe vorbereitet ist. TCP/IP ist nicht ausgelegt für die hohen Anforderungen des VoIP Verkehrs, da es längere Verzögerungen und schwankende Bandbreiten akzeptiert. Im schlechtesten Fall besteht ein solches Netz aus einem Konglomerat aus diversen Hardwarekonfigurationen verschiedener Hersteller, oftmals unzureichend konfiguriert und unpassend dimensioniert.

In dieser Arbeit wird untersucht, wie weit QoS Mechanismen mit der QoS Simulation übereinstimmen, um dadurch den Einsatz von VoIP in Datennetzen simulieren zu können.

Die QoS Mechanismen ermöglichen eine gewisse Steuerung des Datenverkehrs durch ein Bandbreitenmanagement und verschiedene Queueing Mechanismen. Vereinfacht ermöglichen sie den Datenpaketen des VoIP Datenstromes eine bevorzugte Behandlung und vermindern so Staus und Engpässe.

Als Verifizierung des Berkeley NS-2 Simulators wird ein Experiment herangezogen, bei welchem eine Simulation im Labor mit realer Hardware nachgestellt wird und die Ergebnisse verglichen werden.

Um die QoS Mechanismen hinsichtlich ihrer Wirkungsweise vergleichen zu können, werden diese mittels einer geeigneten Teststellung mit dem Berkeley NS-2 Simulator simuliert und die Resultate mit einer Simulation ohne QoS verglichen.

Der Berkeley NS-2 Simulator liefert realistische Aussagen über bestehende Computernetzwerke für VoIP Datenverkehr. Darüber hinaus ermöglicht er die spezielle Untersuchung und Simulation mittels seiner implementierten QoS Mechanismen.

Die Grundlagen und Eigenschaften der VoIP und TCP/IP Protokolle, QoS Mechanismen und die Funktion des Berkeley NS-2 Simulators werden mit recherchierter Literatur beschrieben. Der Berkeley NS-2 Simulator wird mittels einer geeigneten Simulation im Vergleich zu einem echten

Computernetzwerk verifiziert. Um die QoS Mechanismen zu testen, wird mit Hilfe des Berkeley NS-2 Simulators eine definierte Teststellung mit und ohne diesen simuliert.

Ziel der Arbeit ist die Simulation verschiedener QoS Systeme, um allgemeine Aussagen über deren Funktionalität in Bezug auf VoIP oder ähnliche Echtzeitanwendungen in großen Computernetzen zu ermöglichen.

1. Das TCP/IP Netzwerkprotokoll

Mitte der 70er Jahre begann die ARPA (*Advanced Research Projects Agency*) mit dem Entwurf einer Netzwerk Technologie, die zwischen 1977 und 1979 einen technischen Level erreicht hat, der auch heute noch Gültigkeit hat. Das Resultat war das ARPANET, welches im Jahre 1980 die Ära des globalen Internet einleiten sollte. Zu dieser Zeit begann die ARPA ihre Rechner auf das *Transmission Control Protocol / Internet Protocol* TCP/IP umzustellen.

In weiterer Folge trennte man die vom bisherigen TCP (*Transmission Control Protocol*) erledigten Aufgaben in mehrere Protokolle. Für die Adressierung zuständig wurde das IP Protokoll und die Datenfluss-steuerung übernahm das TCP Protokoll. Als »unsicheres« Protokoll konnte anstelle von TCP auf das UDP (*User Datagram Protocol*) Protokoll zurückgegriffen werden. (vgl. LINUXFIBEL, Die Geschichte des Internet, 2005)

1.1. IP – Internet Protocol

Bei TCP/IP übernimmt im Wesentlichen das *Internet Protocol* (IP) die Kontrolle. Das IP Protokoll wird im RFC 791 (*Request for Comment*) spezifiziert und erfüllt zwei wichtige Grundfunktionen: Einerseits regelt es die Adressierung der Daten, andererseits deren Fragmentierung. Jedes *Datagram*, auch als Datenpaket bezeichnet, wird mit einem IP-*Header* versehen. Bei der Adressierung werden im *Header* die *Source*- und die *Destination*-Adresse, also Ursprung und Ziel angegeben. Dazu werden 32 Bit lange Adressen verwendet (IP-Adressen). Diese ermöglichen die Unterscheidung von ca. 4 Milliarden Rechnern (vgl. RFC

791, 1981, S.7). Trotz der großen Zahl der möglichen Adressierungen wird in naher Zukunft dieser Adressenraum nicht mehr ausreichend sein. Deshalb ist IPv6 (*Internet Protocol Version 6*) entwickelt worden, welches auf einen 128 Bit breiten Adressenraum zurückgreift.

Da die maximale Größe der Datenpakete 65 535 Bytes beträgt, dies aber zu groß für die meisten Netzwerke ist, entstehen aus den Rohpaketen viele kleine Fragmente, meist kleiner als 1518 Bytes (vgl. WIKIPEDIA, 2005, IPv4). Diese werden einzeln übertragen und beim Empfänger wieder zu einem Paket zusammengesetzt. Um diese Funktion zu gewährleisten, sind in jedem *Header* der Fragmente Informationen enthalten: Es gibt das Feld *Flags*, welches über 3 Bit steuert ob ein Paket geteilt worden ist. Dazu gibt es das 13 Bit lange *Fragment Offset*, welches die Fragmente nummeriert, damit sie wieder in der richtigen Reihenfolge zusammengesetzt werden können.

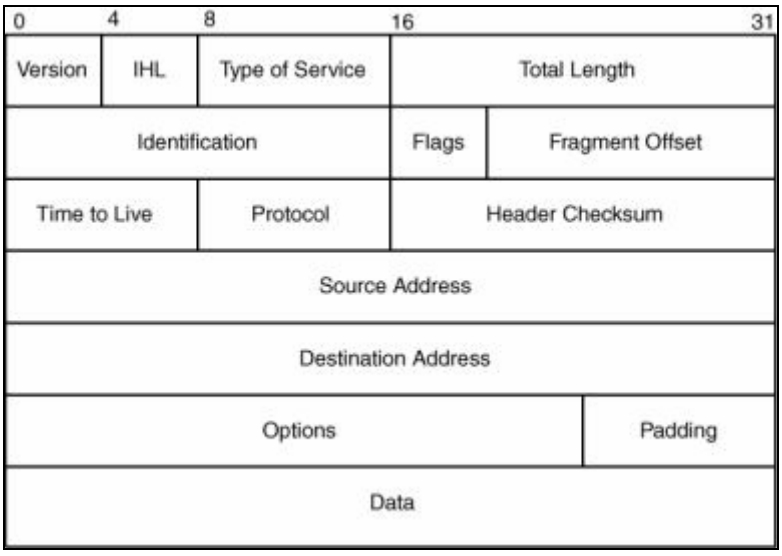

Abbildung 1: IP Header (www.syngress.com/book_catalog/112_ipsec/112_CiscoSec_toce_01_files/image017.jpg)

Grundsätzlich ist IP ein verbindungsloses Protokoll, und daher wird jedes Paket völlig unabhängig behandelt. Die entstehenden Laufzeitunterschiede, durch die verschieden schnell eintreffenden Pakete, können bei VoIP Übertragungen durch einen Jitterpuffer korrigiert werden (Siehe Kap. 2).

Die Pakete werden vom Empfänger nicht bestätigt, so können diese auf dem Weg verworfen werden, bzw. doppelt ankommen oder vom Empfänger nicht angenommen werden. Ein weiterer Schwachpunkt ist, dass keine Fehlerkontrolle existiert, außer die Möglichkeit der Bitfehlerüberprüfung mittels der *Header Checksum* (=Prüfsumme) des IP *Headers*.

Damit der Netzwerkverkehr reibungslos funktioniert, wurden Transportprotokolle entwickelt, welche auf IP aufbauen und versuchen dessen Schwachstellen ausgleichen, wie z.B. das RTP Protokoll für Sprachübertragungen. (vgl. TANENBAUM, 2003, S.432ff; vgl.TRUMMER, 2001, S.10ff)

1.2. UDP – User Datagram Protocol

Im RFC 768 geregelt, baut dieses Protokoll auf dem IP Protokoll auf und erweitert es um zusätzliche Funktionen. Zur Vermeidung von zusätzlichen Verwaltungsdaten, in weiterer Folge als *Overhead* bezeichnet, beschränkt man sich auf vier zusätzliche Felder im *Header*: Die ersten zwei sind der *Source Port* (=Quellport) und der *Destination Port* (=Zielport). Diese 16 Bit breiten Felder ermöglichen die Adressierung von 65.536 verschiedenen Ports. So können mehrere Applikationen auf einem Rechner simultan Pakete entgegennehmen, vorausgesetzt jedes nutzt einen eigenen Port. Mit der Angabe des *Destination Port* kann beim Senden eines Paketes die Applikation ausgewählt werden, die es empfangen soll. Für bidirektionale

Kommunikation steht der *Source Port* zur Verfügung, auf dem der Sender auf die gewünschte Antwort wartet. Ein weiteres Feld ist *Length* (=Länge), welches die Anzahl der Bytes des gesamten Paketes enthält (*Header* und Nutzdaten). Im letzten Feld des *Headers* steht dann noch die *Checksum* (=Prüfsumme). Diese enthält die Prüfsumme der Nutzdaten.

UDP ist kein übertragungssicheres Protokoll, da es vorkommen kann, dass ein Paket doppelt ankommt oder verloren geht. Es wird oftmals für Echtzeitübertragungen wie VoIP genutzt, da es mit wenig *Overhead* auskommt und Fehler in Kauf genommen werden. (vgl. STALLINGS, 1997 S. 619ff; vgl. TANENBAUM, 2003, S.524ff; vgl. TRUMMER, 2001, S.13ff).

1.3. TCP – Transmission Control Protocol

Das Transmission Control Protocol (TCP), welches im RFC 793 spezifiziert wird, stellt ein alternatives Transportprotokoll zu UDP dar. Anders als dort, handelt es sich bei TCP um ein zuverlässiges Protokoll, da es doppelte oder fehlende Elemente verhindert und dadurch eine garantierte Übertragung gewährleistet.

Ebenso wie im UDP *Header*, gibt es je ein Feld für den S*ource* und den *Destination Port*, sowie eine *Checksum*. Zusätzlich existieren noch weitere Felder, die für den Verbindungsaufbau und die Datenübertragung benötigt werden.

Jedes TCP Paket wird mit einer Sequenznummer ausgestattet, die dem Empfänger die Erkennung von doppelten, fehlenden oder vertauschten Paketen ermöglicht. Dabei nutzt das TCP Protokoll das so genannte *Sliding Window* Verfahren, basierend auf Sequenznummern. Dazu muss der Empfänger zuerst wissen, mit welchem Paket bzw. welcher

Sequenznummer die Übertragung beginnt. Es synchronisieren sich Sender und Empfänger mit Hilfe eines Drei-Wege-Handshakes zu Beginn, um danach eine zuverlässige Übertragung zu garantieren.

Die *Window Size* ist die Anzahl der Pakete, welche übertragen werden, ohne auf eine Empfangsbestätigung zu warten (*Acknowledge* (ACK)).

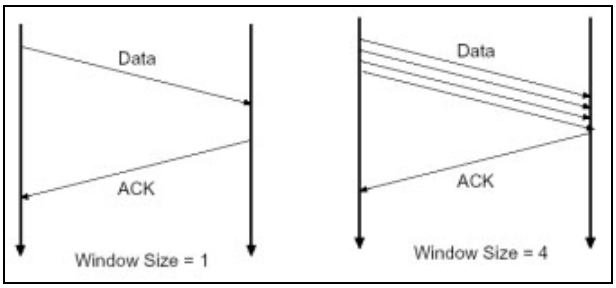

Abbildung 2 Window Size (www.soi.wide.ad.jp)

Nachdem der Sender die Empfangsbestätigung für die Pakete erhalten hat, verschiebt er sein Fenster und sendet die nächsten Pakete.

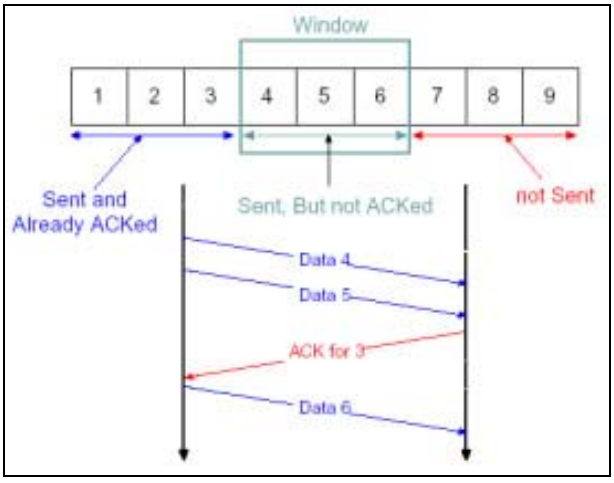

Abbildung 3 Sliding Window (www.soi.wide.ad.jp)

Der *Sliding Window* Mechanismus ermöglicht dem TCP Protokoll eine flexible Flussteuerung. Der Empfänger kann so je nach Situation dem Sender eine kleinere oder größere *Window Size* vorgeben um so möglichst effizient die Bandbreite zu nutzen.

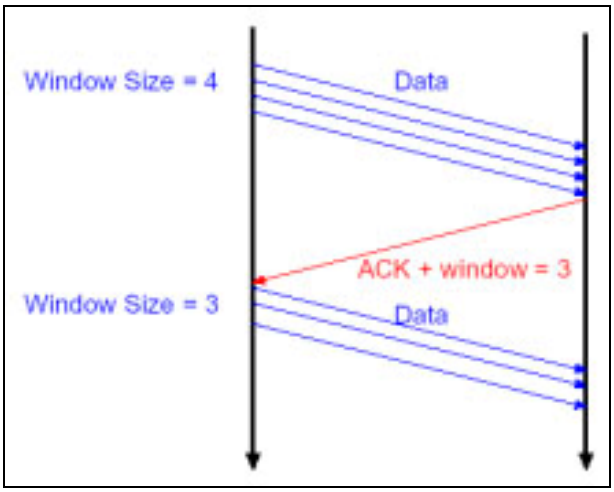

Abbildung 4 Window Size 2 (www.soi.wide.ad.jp)

(vgl. STALLINGS, 1997, S. 610ff, vgl. TANENBAUM S.532ff; vgl. TRUMMER, 2001, S.13ff).

2. VoIP Grundlagen und Funktionalität

2.1. Grundlagen

Bereits Anfang der neunziger Jahre wurde an der Übertragung von Sprachdaten über IP –Netze geforscht und erste zuverlässige Software entwickelt. So ermöglichte die Software „Phone Talk“ bereits 1990 die Telefonie über lokale IP-Netze. Im Jahre 1995 präsentierte das israelische Unternehmen VocalTec Communications Ltd. (vgl. VOCALTEC, 2005) eine der ersten Lösungen für Sprachübertragungen über das Internet. Die Zukunftschancen dieser Technologie werden als außerordentlich hoch eingeschätzt, verspricht man sich doch von der Zusammenführung der Daten- und Telefonnetze große Einsparungen bei Kosten und Administration. Doch die Konvergenz reiner Computerdaten und Sprachdaten erfordert zuverlässige Technologien und Protokolle, um den Anforderungen beider Anwendungen gerecht zu werden.

Die wichtigsten Aspekte sind:

- Art der Sprachkodierung und Komprimierung der Sprachdaten
- Bandbreitenanforderungen
- Verzögerungen / Laufzeiten auf dem Übertragungsweg
- Übertragungsgüten wie CoS (*Class of Service*) / QoS (*Quality of Service)* / IP – TOS / MPLS (*Multi Protocol Label Switching*)

(vgl. KÖHLER, 2002, S.93)

Als CoS bezeichnet man eine Gruppe von Verfahren zur Priorisierung in TCP/IP-basierten Netzwerken, die in IEEE 802.1p standardisiert sind. CoS ermöglicht eine gezielte Priorisierung, während mit QoS explizite Bandbreitengarantien oder -beschränkungen eingerichtet werden.

Grundsätzlich funktionieren alle VoIP Systeme ähnlich. Die zunächst analog vorliegende Sprache wird mit Hilfe eines A/D (Analog/Digital) Wandlers digitalisiert und anschließend komprimiert. Das am weitesten verbreitete Verfahren hierfür ist das PCM (*Pulse Code Modulation*) Verfahren, welches von ITU (*Institute of Electronical and Electronics Engineers*) in der G.711 Spezifikation standardisiert wurde. Danach wird das Signal paketiert und über jedes beliebige IP-Netz übertragen. Beim Empfänger angekommen, werden die Signale wieder zusammengesetzt, dekomprimiert und in analoge Sprache gewandelt.

Die Schwierigkeit dabei ist einerseits das paketorientierte TCP/IP System, andererseits ein bis dato immer verbindungsorientierter Dienst wie Sprachübertragung. Bei der Qualität der Sprachübertragung spielt die Kodierung der Sprache eine wichtige Rolle. (vgl. KÖHLER, 2002, S.94).

2.1.1. Sprachkodierung

Ein wesentliches Element ist die Art der Kodierung. Hier muss ein Kompromiss gefunden werden zwischen der benötigten Bandbreite, der Sprachqualität und der Dauer der Kodierung. Um hohe Qualität und geringe Bandbreite zu erreichen, werden starke Kompressionsverfahren, welche aber sehr rechenaufwändig sind, benötigt, die aber *Delays* (Verzögerungen) mit sich bringen.

Kompressionsmethode	Datenrate [Kbit/s]	zus. Verzögerung [ms]
G.711	64	0,75
G.726 ADPCM	32	1
G.728 LD-CELP	16	3-5
G.729 CS-ACELP	8	10
G.729a CS-ACELP	8	10
G.723.1 MPMLQ	6,3	30
G.723.1 ACELP	5,3	30

Abbildung 5: Sprachkodierung (MEINCKE, 2000)

In dieser Grafik orange unterlegt sind die LPC (*Linear Predictive Codes*) für höhere Komprimierungsraten. Diese werden durch blockweise Verarbeitung bei festen Blockgrößen von ca. 10-20 ms (je nach Codec/Modus) erreicht. Die erste Kodierung G.711 ist die unkomprimierte ISDN-Kodierung mit 8 Bit. Die G.726 Kodierung ist eine Differenzkodierung mit 4, 5 oder 6 Bit (vgl. MEINCKE, 2000, S.13, vgl. LUTZ, 2002 S.9f; vgl. KÖHLER, 2002, S.93ff).

2.1.2. Laufzeitunterschiede

Da Verzögerungen in einem Netzwerk durch die Dauer der Paketübertragung nicht verhindert werden können, gilt es sie zu minimieren, wobei ein Hauptaugenmerk auf dem sogenannten *Jitter* (Laufzeitunterschied) liegt, der durch das verschieden schnelle Einlangen der Datenpakete bedingt ist. Wie im vorigen Kapitel erläutert, sortiert das TCP Protokoll auf der Empfängerseite die Pakete wieder in die richtige Reihenfolge, was einen gewissen „Jitter-Puffer“ voraussetzt. Dieser kann die Pakete bis zu 30 ms zurückhalten. Wird der Speicher zu klein gewählt, ist es möglich, dass das System nicht genug Zeit hat, auf alle Pakte zu warten, bzw. diese richtig zu sortieren. Dies hat zur Folge, dass die Sprache unverständlich wird. Andererseits kann ein zu großer Jitter-Puffer unnötige Verzögerung erzeugen. Es ist daher üblich,

dynamische „Jitter-Puffer“ verwendet, die ihre Größe dem momentanen Bedarf angleichen (vgl. HOGREFEL, 2004, S.40ff (Teil2)).

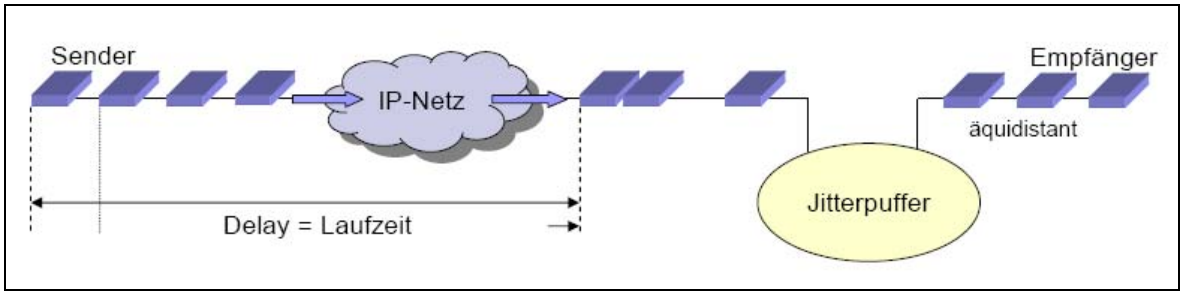

Abbildung 6: Jitterpuffer (Hogrefe, Mobilkommunikation 2 Teil2)

In der Regel wird für Sprachübertragungen das UDP Protokoll verwendet. Dieses fordert im Gegensatz zum TCP Protokoll verlorene Pakete nicht neu an, da dieser Vorgang bei der Sprachübertragung zu lange dauern würde. Bis zu einem Verlust von 5% der Sprachdaten kompensiert der Mensch automatisch die fehlenden Informationen durch die Redundanz der Sprache. Darüber liegende Fehlerraten werden als inakzeptable Sprachqualität wahrgenommen (vgl. LUTZ, 2002, S.10f; KÖHLER, 2002, S.101ff).

2.1.3. Delay

Als *Delay* wird die Dauer vom Zeitpunkt des Absendens bis zum Einlangen beim Empfänger bezeichnet, wobei hauptsächlich die Länge und Geschwindigkeit der Strecke, deren Auslastung und die dazwischen liegende Hardware ausschlaggebende Faktoren sind. Jede aktive Netzwerkhardware verursacht technisch bedingte zusätzliche Verzögerungen zu den bereits vorhandenen distanzbedingten *Delays*.

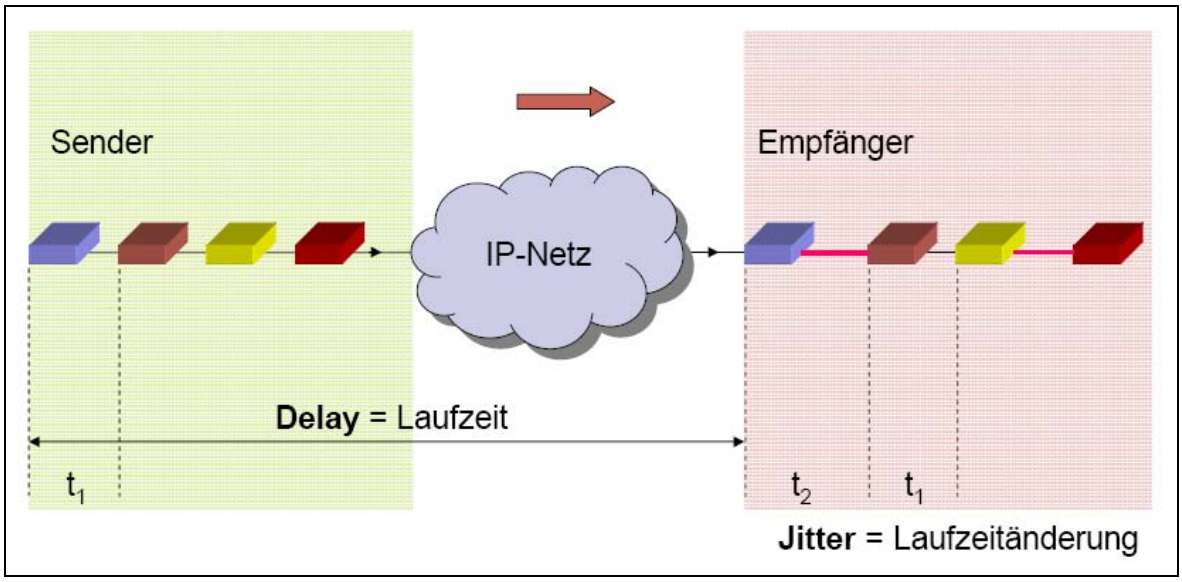

Abbildung 7: Delay (Hogrefe, Mobilkommunikation 2 Teil2)

2.1.4. Gesamtlaufzeit

Im Endeffekt ist nur die Gesamtlaufzeit relevant, die Zeit zwischen dem Mikrofon des einen und dem Lautsprecher des anderen Gesprächsteilnehmers. Sie setzt sich zusammen aus den Audiocodec-Verzögerungen, den Laufzeitverzögerungen und dem Jitter-Puffer.

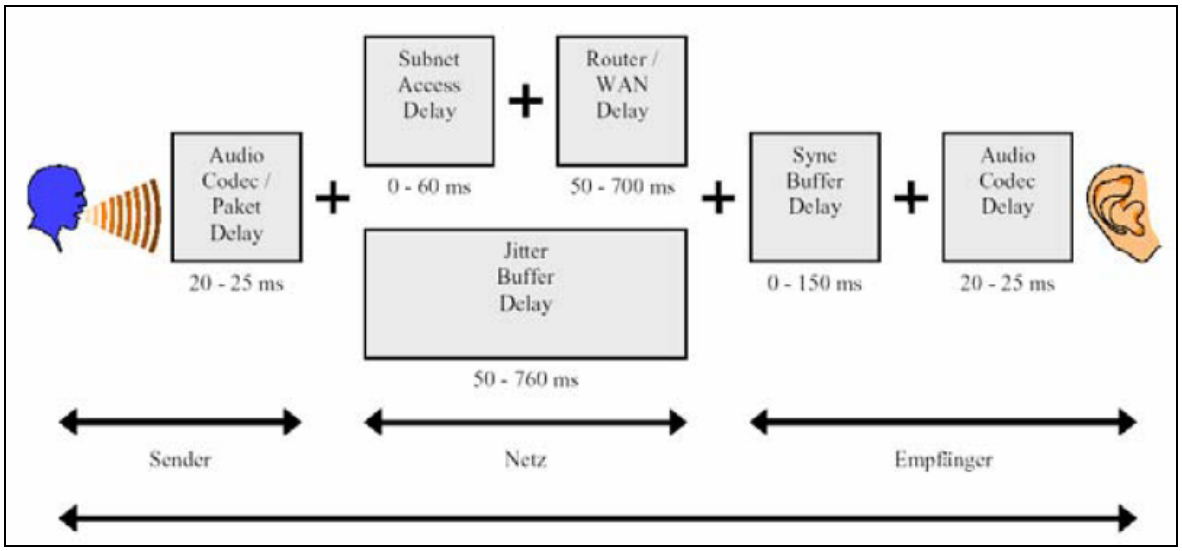

Abbildung 8: Gesamtlaufzeit (Hogrefe, Mobilkommunikation 2 Teil2)

Für diese Gesamtlaufzeit hat die ITU-T (*International Telekommunikation Union–Telecommunications Standardization Sector*) in der Empfehlung G.114 die Grenzen für die Akzeptanz festgelegt. Alles unter 150 ms ermöglicht sehr gute Sprachqualität, alles über 400 ms gilt als unakzeptabel.

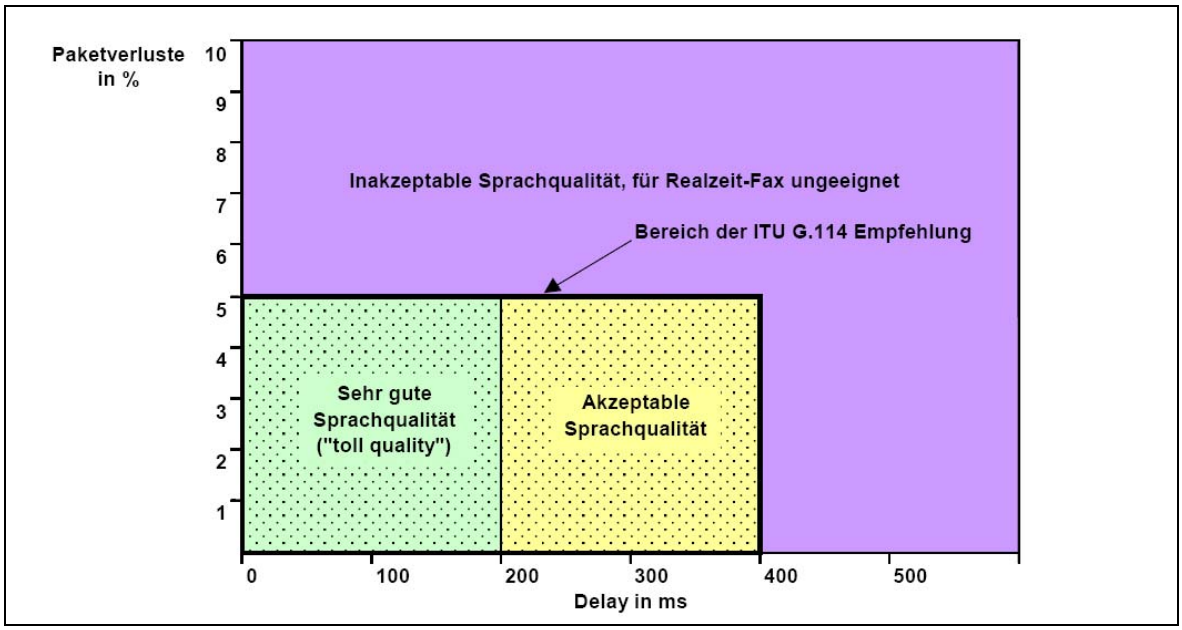

Abbildung 9: Empfehlung G.114 der ITU-T (Hogrefe, Mobilkommunikation 2)

2.1.5. Bandbreitenanforderungen

Je nach Kodierung stellen die Sprachübertragung und deren Protokolle gewisse Bandbreitenanforderungen an das Netzwerk. Bei der IP Telefonie wird die Gesprächsbandbreite mit 64 Kbit/s (unkomprimiert, ISDN) gemessen, was ungefähr 150.000 simultanen Gesprächen über einen 10 Gbit/s Backbone ermöglichen würde. Bei einer Komprimierung auf 12 Kbit/s wären schon ca. 800.000 Gespräche möglich, bei 8 Kbit/s sogar 1,25 Millionen.

Für VoIP-Anwendungen wird meistens ein Codec der G.729xx Spezifikationsserie der ITU-T verwendet und bildet damit die Grundlage

für eine normgerechte Sprachqualität. Unter Berücksichtigung des *Overheads* und der Sprachkomprimierung wird eine Bandbreite von ca. 10 kbit/s (1,25 kByte/s) pro Sprachverbindung benötigt. Diese Bandbreite muss das Datennetz für jedes Gespräch gewährleisten. (vgl. KÖHLER, 2002, S.95).

2.2. Protokolle und Komponenten

2.2.1. H.323

Um reibungslos weltweit Audio- und Videodaten übertragen zu können hat die ITU-T 1996 die Standards der H.32x Serie herausgebracht. Ziel war es, herstellerunabhängig und interoperabel mit bestehenden Systemen, sowohl Festnetz- als auch Mobilfunksystemen, zu agieren.

Dafür wurden 5 Hauptstandards festgelegt:

- H.320 für ISDN (*Integrated Services Digital Network*)
- H.321 und H.310 für B (*Broadband*) – ISDN
- H.322 für LAN (*Local Area Network*) mit QoS Garantien
- H.323 für LAN ohne QoS Garantien
- H.324 für SCN (*Switched Circuit Network*)

(vgl. KÖHLER, 2002, S.140)

H.323 Terminal

Als Terminal wird ein (multimedialer) Endpunkt zur Kommunikation bezeichnet, was bei VoIP ein Telefon bedeutet. Dies kann in zwei Varianten auftreten, als sogenanntes „*Softphone“* (bzw. *Softclient*), wenn ein Telefon mittels Software emuliert wird, oder als „echtes“ IP-

Telefon mit konventioneller Telefonausstattung. Ein Beispiel für ein *Softphones* ist z.B. Microsofts Netmeeting.

H.323 Gateway

Gateways sind ebenfalls Endgeräte, allerdings dienen diese als Schnittstelle zu anderen Systemen, wie etwa zum herkömmlichen Telefonnetz. Die Hauptfunktion eines Gateways besteht demzufolge in der Vermittlung zwischen dem Telefonnetzwerk und dem IP Netzwerk. Dabei managt es den Auf- und Abbau der einzelnen Sprachkanäle sowie die Umsetzung der Sprachdaten, und die unterschiedlichen Signalisierungen der Vermittlungsprotokolle. Ein Gateway kann z.B. das weit verbreitete SS7 Signalisierungsprotokoll des ISDN auf die entsprechende Signalisierung eines VoIP Netzes umsetzen.

H.323 Gatekeeper

Als zentrale Stelle im H.323 Netzwerk verwaltet ein *Gatekeeper* die Kommunikationsbeziehungen. Er ist optional, kann aber auch mehrfach vorhanden sein. Er erledigt den Rufauf- und abbau zwischen den kommunizierenden H.323 Terminals. Dazu registriert er angemeldete Endgeräte mit ihren Rufnummern oder *Aliases* (Verknüpfungen) und wandelt diese zu IP-Adressen bzw. Domainnamen um (vgl. KÖHLER, 2002, S 140ff). Weitere wichtige Funktionen der *Gatekeepers* sind das Verwalten der Zugriffsrechte der einzelnen Teilnehmer, Bandbreiten-management und das Aufzeichnen der Verbindungsdaten z.B. für Abrechnungen. Falls mehrere Gatekeeper in einem Netz existieren, wird das Netz in sogenannte „Zonen“ unterteilt, welche jeweils von einem *Gatekeeper* gemanagt werden.

H.323 Multi Point Unit

Die Multi Point Unit ermöglicht Konferenzschaltungen zwischen mehreren Teilnehmern, wobei sie wie ein Terminal agiert, welches in der Lage ist, die Gespräche von mehreren Anrufern, bzw. Konferenzteilnehmern, entgegenzunehmen und zu halten.

Der H.323 Protokollstack

Als multifunktionales Signalisierungsprotokoll enthält der H.323-Stack folgende Komponenten:

- Die G7.xxx Audiocodecs
- Die T.12x-Reihe für Multimedia-Datenkommunikation
- Die Videoübertragungsprotokolle H.261 und H.263
- Das RTP (*Realtime Transport Protocol*) Protokoll zur Echtzeit-Datenübertragung und das dazugehörige Kontrollprotokoll RTCP (*Realtime Transport Control Protocol*)
- Die Signalisierungsprotokolle H.245, H.235, Q.931 und RAS (Remote Access Service) (vgl. LUTZ, 2002, S.13)

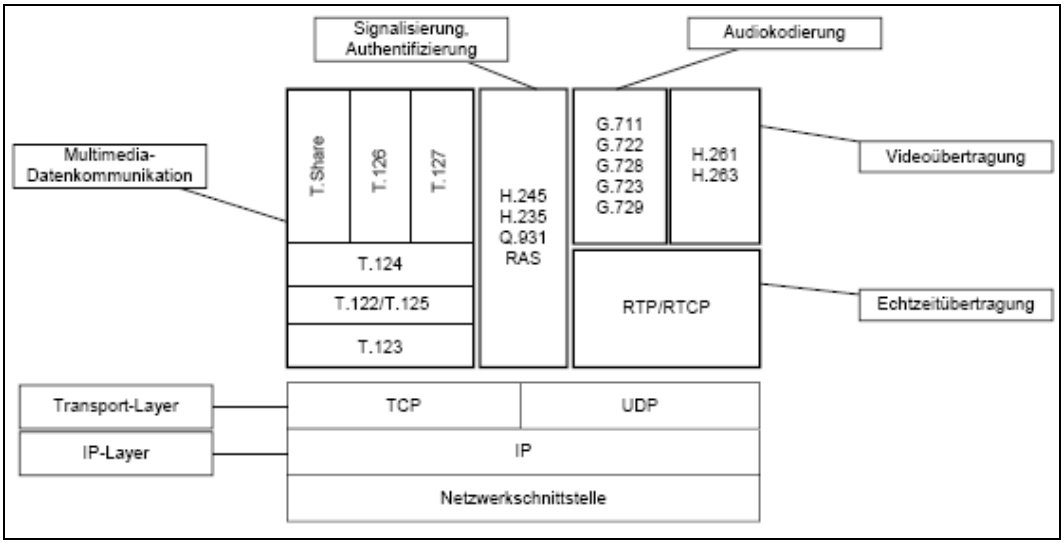

Abbildung 10 Der H.323 Protokollstack (vgl. LUTZ, 2002; S.13)

2.2.2. SIP

SIP (*Session Initiation Protocol*) ist ein sitzungsbasierendes Signalisierungsprotokoll für Internetkonferenzen und Internet Telefonie. Es wurde von der IETF (*Internet Engineering Taskforce*) entwickelt und im RFC 2543, im Jahr 2000, standardisiert. Da es aus der Internetwelt stammt, ist es ein einfaches, offenes und sehr internetfreundliches Protokoll, das speziell für die Initialisierung und das Management von interaktiven Sitzungen, wie z.B. Sprache über WAN (*Wide Area Network*) entwickelt wurde (vlg. KÖHLER, 2002, S.65).

Es ist wie H.323 ein Punkt-zu-Punkt-Protokoll, verwendet ebenfalls RTP für den Pakettransport, reicht aber nicht an dessen Komplexität heran. SIP ist ein reines Signalisierungsprotokoll in der TCP/IP Anwendungsschicht, welches sich in der Regel UDP als Transport-protokoll bedient. Durch seine Herkunft ist das Protokoll ähnlich bzw. angelehnt an bereits bestehende Protokolle, wie z.B. das HTTP Protokoll und DNS (*Domain Name System*).

Die Grundelemente einer SIP Infrastruktur bestehen aus vier Komponenten (vgl. JOHNSTON, S.17ff):

1) User Agents

2) Registrierungsserver

3) Proxyserver

4) Umleitungsserver

User Agent

Bei SIP stellen die User Agents die Kommunikationsendpunkte dar. In der Regel sind das IP-Telefone. Ein User Agent kann sich in zwei verschiedenen Zuständen befinden: Wird vom Agent eine Verbindung zu einem Server aufgebaut, befindet er sich im *Client Mode* und wird als

User Agent Client bezeichnet. Wird er angerufen, bezeichnet man diesen Status als *User Agent Server*, da er für die Gegenstelle als Server fungiert. (vgl. JOHNSTON, S. 43ff).

Registrierungsserver

Dieser Server verwaltet innerhalb seiner Domäne alle angemeldeten User, da sich alle Clients an diesem anmelden müssen, und den Kontakt in regelmäßigen Abständen halten, um die Daten so aktuell wie möglich zu halten. Darüber hinaus bietet der Server die Speicherung von Userpräferenzen, wie etwa die gewünschte Erreichbarkeit (nicht stören etc.) oder persönliche Informationen (vgl. JOHNSTON, S.47ff).

Proxyserver

Ähnlich der Funktion eines H.323 Gatekeepers ist die Aufgabe des Proxyservers, dass die Verbindung zwischen den richtigen Teilnehmern aufgebaut wird, nur ist er einzig auf Clientanfragen ausgelegt (vgl. JOHNSTON, S 47ff).

Umleitungsserver

Die Funktion dieses Servers besteht darin, anfragende User Agents über die aktuelle (wenn neue, temporär andere) Erreichbarkeit anderer User zu informieren. Dabei stellt er selbst keine Verbindungen her, sondern dient nur als aktuelle Datenbank. Die Verbindung stellen die User Agents selbst untereinander her (vgl. JOHNSTON, S. 52ff).

In der technischen Umsetzung werden die verschiedenen Server meist nicht physikalisch getrennt, sondern mittels einer multifunktionalen Software auf einem einzigen Server betrieben.

2.2.3. Vergleich H.323 und SIP

Technisch ermöglichen diese beiden Protokolle eine Übertragung multimedialer Daten über das RTP Protokoll, allerdings mit zwei völlig verschiedenen Ansätzen. Vor allem im Internet befindet sich das jüngere SIP auf dem Vormarsch.

Obwohl beide Protokolle das Telefonieren über IP-Netzwerke ermöglichen, sind sie dennoch im Detail sehr unterschiedlich, da sie sich bereits bei der Kodierung unterscheiden. Das SIP Protokoll kommuniziert als rein textbasierendes Protokoll über formatierte Textmeldungen im Klartext, wohingegen das H.323 Protokoll durch binärcodierte, relativ lange Signalisierungscodes arbeitet (vgl. JOHNSTON, S. 187f).

Es ist erkennbar, dass das H.323 Protokoll aus der Signalisierung leitungsvermittelnder Dienste wie ISDN (Q.931) stammt und dementsprechend umfangreicher aufgebaut ist, im Gegensatz zu dem relativ einfachen, an das HTTP Protokoll angelehnte, SIP Protokoll.

Die nächste Tabelle stellt die dadurch resultierenden Vor- und Nachteile überblicksmäßig dar.

	H.323	SIP
Vorteile	• Sehr genaue Spezifikationen • Lastverteilung möglich • weit verbreitet und etabliert	• optimal an das Internet angepasst • leicht administrierbar (textbasierend) • für mobile Geräte geeignet • Verteilen auf mehrere Anschlüsse möglich („forking“)
Nachteile	• relativ langsamer Aufbau • sehr komplex	• weiterführende Dienste wie Verrechnung kompliziert zu realisieren • kompliziert bei der Verwendung von NAT (*Network Address Translation*) • im Netzwerk ohne Internetbasisdienste wie DNS oder Proxys nicht lauffähig

Abbildung 11 Vergleich H.323 & SIP

2.2.4. Marktübersicht

Es existiert bereits eine große Anzahl an verschiedenen, sowohl kostenpflichtigen, als auch kostenfreien Programmen, um H.323 oder SIP Lösungen zu steuern. Die weit verbreitete und kostenlose Asterisk Software Suite bietet zum Beispiel eine große Auswahl an Protokollen

und Diensten, ebenso wie auf der kostenpflichtigen Seite Lösungen von großen Anbietern wie Cisco mit dem Callmanger, bereitgestellt werden.

2.3. RTP – Realtime Transport Protocol

Schon lange bevor das Internet entstand, zu Zeiten des ARPANET, wurde mit der Übertragung von Audiodaten experimentiert. Diese Art der Übertragung wurde als Realtime klassifiziert und erforderte schon damals, wie im Kapitel „VoIP Grundlagen“ erwähnt, eine konstante Übertragungsrate und möglichst geringe Verzögerungen.

Deshalb ist für IP Netze ein Protokoll nötig, welches steuernd eingreift um Realtime Übertragungen zu ermöglichen – das RTP Protokoll. Bei diesem wird statistisches Multiplexing angewandt, das mit unterschiedlich großen Paketen arbeitet, da Router für jede Netzwerkschnittstelle nur immer ein Paket auf einmal übertragen können. Dies bedeutet, dass der Übertragungszeitpunkt eines Pakets auch von der Größe des vorherigen Pakets abhängt. Im RFC 1889 definiert, regelt es in erster Linie die Bearbeitung von vertauschten bzw. verdoppelten Paketen (vgl. TANENBAUM, 2003, S. 529ff).

2.3.1. Eigenschaften von RTP

Das RTP Protokoll verwendet für die Übertragungen zwei Kanäle: Im ersten Kanal findet die eigentliche Übertragung der Nutzdaten statt. Der zweite Kanal dient der Übertragung von Kontroll- und Statusinformationen über das RTCP (RealTime Control Protocol) Protokoll. Die wesentlichen Aufgaben dieses Steuerungsprotokolls sind die Generierung und Auswertung von Rückmeldungen der Gegenstelle. Diese sind nötig, um die verschiedenen Parameter der aktuellen Übertragungsqualität anzupassen. So wird z.B. die Größe des

Jitterpuffers der aktuellen minimalen und maximalen Verzögerung angepasst, um Aussetzer zu vermeiden.

Um ein Telefonat aufzubauen, müssen die Teilnehmer sich zunächst auf eine Multicast-Adresse und zwei Port-Nummern für die RTP- und RTCP-Pakete einigen.

Während des Telefonates verpackt jeder Teilnehmer den von ihm erzeugten Datenstrom in RTP-Pakete fester Größe. Diese werden an die vorher ausgehandelte Multicast-Adresse geschickt und vom Netzwerk an alle Multicast Teilnehmer weitergeleitet. Anhand der Zusatz-informationen, die jedes RTP Paket neben den Nutzdaten enthält, bringt der Empfänger die Pakete in die richtige Reihenfolge, synchronisiert sie, identifiziert das gekapselte Datenformat und rekonstruiert die Nutzdaten für die Wiedergabe. Jeder teilnehmende Sender wird dabei im RTP Paket über eine eindeutige Kennung (SSRC, (*Synchronization Source*) identifiziert.

Obwohl RTP auf UDP basiert und dessen Eigenschaften übernimmt, ist es möglich auch andere Transportprotokolle, welche über Multiplexing-mechanismen verfügen um RTP- und RTCP- Pakete getrennt zu übertragen, zu verwenden (vgl. SCHULZRINNE, CASNER, FREDERICK, JACOBSON, 2001, Kap. 10).

2.3.2. Aufbau

Die RTP Pakete werden mit einem RTP *Header* bestückt. Dieser beginnt mit einer 2 Bit breiten Versionsnummer. Danach folgen die P (*Padding*) und X (*Extension*) Bits. Ein gesetztes P-Bit bedeutet, dass die Nutzdaten mit 0-Bytes gefüllt sind, was manche Verschlüsselungsverfahren wegen

bestimmter Blockgrößen benötigen. Das X-Bit zeigt an, ob ein zusätzlicher *Header* vorhanden ist. Je nach Anwendung kann solch ein zusätzlicher spezifischer *Header* in die Nutzdaten integriert werden Das CC (*Contributing Source Identification Count*) enthält die Anzahl der verschiedenen Quellen bei gemischten RTP-Strömen. Im M Feld kann ein *Marker* zur Identifikation verschiedener Profile gesetzt werden. Darüber hinaus wird noch der *Payload Type* (PT), die Art der übertragenen Nutzdaten, angegeben um die Daten richtig interpretieren zu können (vgl. TANENBAUM, 2003, S.531).

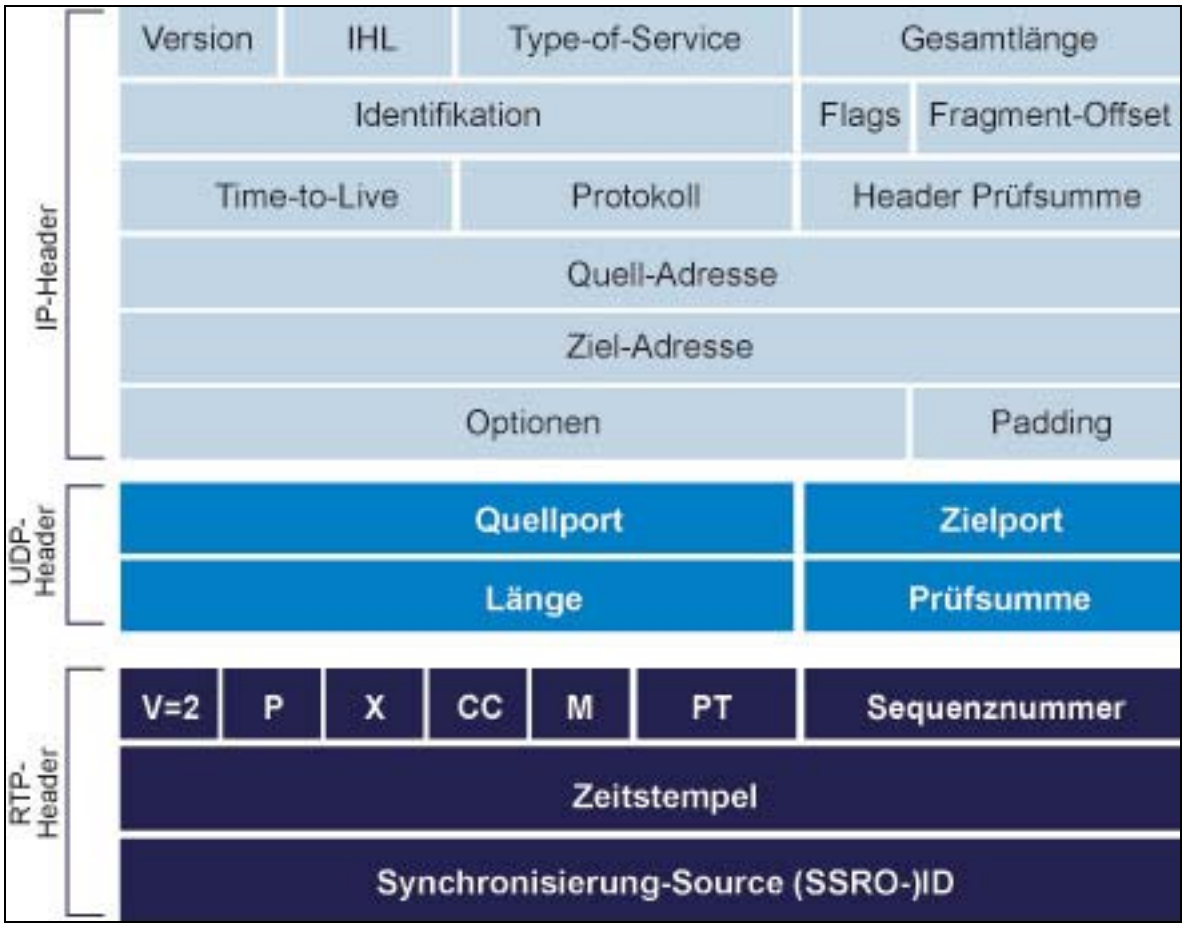

Abbildung 12 RTP Header
(www.tecchannel.de/netzwerk/grundlagen/431385/index5.html)

Dahinter folgt die *Sequence Number*, welche der fortlaufenden Nummerierung dient, um vertauschte Pakete wieder richtig zusammen

zu setzen. Aus dieser Tatsache und der, dass Pakete verloren gehen können und der Jitter-Puffer Pakete zurückhält, resultiert das Problem, dass der zeitliche Zusammenhang verloren gehen kann. Daher wird jedes Paket mit einem Zeitstempel versehen um die Pakete auch in der richtigen zeitlichen Abfolge wieder zusammensetzen zu können. Um eine Unterscheidung bei mehreren RTP Quellen zu ermöglichen, wird das SSRO (*Synchronization Source*) Feld mit einer Zufallszahl beschrieben (vgl. TANENBAUM, 2003, S.531).

2.3.3. RTP Mixer und Translatoren

Das RTP Protokoll hat zwei Schwachstellen, die bei Konferenzen mit mehreren Teilnehmern zu Tage treten: Einerseits sieht RTP vor, dass alle Teilnehmer die Multimediadaten im gleichen Format senden und empfangen, andererseits entstehen dadurch noch Probleme, dass die RTP Pakete oft von Firewalls geblockt werden (vgl. SCHULZRINNE, CASNER, FREDERICK, JACOBSON, 2001, Kap. 2.3). Diese Probleme werden mit den RTP Mixern und RTP Translatoren beseitigt.

RTP Mixer

Wenn z.B. einer Konferenz mehrerer Teilnehmer in einem schnellen Netzwerk noch ein externer Teilnehmer zugeschaltet wird, welcher über eine qualitativ schlechtere Verbindung angebunden ist, entsteht das Problem, dass dieser über eine deutlich geringere Bandbreite verfügt und somit auf bandbreitenschonendere Audiocodecs angewiesen ist. Dies würde aber bedeuten, dass alle Teilnehmer sich diesem anpassen müssten und ebenfalls den qualitativ schlechteren, hoch komprimierenden Audiocodec verwenden müssten, damit alle Teilnehmer dasselbe Format benutzen. Um den schnell angebundenen Teilnehmern eine

adäquate Verbindung zu ermöglichen, kann in der Nähe des qualitativ schwächeren Nutzers ein RTP Mixer installiert werden, welcher das von den anderen Netzwerk Teilnehmrn verwendete, hochwertigere und ressourcenlastigere Format in ein für den externen Teilnehmer passenderes und bandbreitenschonenderes Format umsetzt (vgl. SCHULZRINNE, CASNER, FREDERICK, JACOBSON, 2001, Kap. 7.3).

RTP Translator

Die meisten Firewalls sperren aus Sicherheitsgründen UDP Pakete, welche aber RTP in der Regel verwendet. Erschwerend kommt dazu, dass die verwendeten Portnummern nicht notwendigerweise festgelegt sind. Eine Möglichkeit wäre die so genannte *Stateful Inspection*, wobei die Firewall jedes UDP Paket analysiert und die RTP enthaltenden Pakete durchlässt. Diese Methode ist rechenaufwändig für die Firewall, weshalb sie selten Verwendung findet; gerade bei hohem Netzwerkverkehr würde dies zusätzliche Verzögerung hervorrufen.

Für dieses Problem wurden die RTP Translatoren entwickelt. Sie werden auf beiden Seiten der Firewall positioniert und nehmen die RTP enthaltenen UDP Pakete vor der Firewall aus dem Datenstrom. Diese werden dann über einen sicheren Kommunikationskanal über die Firewall gesendet. So wird eine Brücke über die Firewall geschlagen, welche die Firewall entlastet und aufwendige Konfigurationen vermeidet (vgl. SCHULZRINNE, CASNER, FREDERICK, JACOBSON, 2001, Kap. 7.2).

3. QoS Mechanismen

Die ITU-T beschreibt QoS wie folgt: „Dienstgüte beschreibt die vollständige Wirkung der Leistungsfähigkeit eines Dienstes, die den Grad der Zufriedenstellung eines Benutzers desselben Dienstes bestimmt“ (STILLER, 1995, S.13).

Im Endeffekt bedeutet dies das reibungslose Funktionieren der gewünschten Applikation, in diesem Fall VoIP. Da Pakete während der Übertragung vertauscht, verzögert, oder verloren werden können, muss um eine zuverlässige und fehlerfreie Übertragung sicherzustellen, ein entsprechendes Transportprotokoll, wie z.B. TCP, verwendet werden.

Alle gerouteten Netzwerke, also das Internet, aber auch große LAN Netze arbeiten nach dem „*Best-Effort*“ Prinzip, wo versucht wird, für jedes Paket den besten Weg zu finden. Nicht berücksichtigt werden die Anforderungen des VoIP Verkehrs, welcher eine maximale Verzögerungszeit (<400 ms, siehe Kap. 2.1.4. Gesamtlaufzeit), und eine feste Mindestbandbreite (10 kbit/s, siehe Kap. 2.1.5. Bandbreiten-anforderungen) benötigt. Diese dürfen auf keinen Fall unterschritten werden, da sonst das Gespräch abgehackt wäre, oder im schlimmsten Fall die Verbindung zusammenbrechen würde.

Dieser Abschnitt widmet sich daher den QoS Philosophien und Netzwerktechnologien, die QoS als festen Bestandteil integrieren. den RSVP (*Resource Reservation Protocol*) und RTP (*Realtime Transport Protocol*) Protokollen.

RSVP ermöglicht eine Art Reservierung von Ressourcen für die Datenübertragung zweier Rechner in einem IP-Netz. RTP stellt als

Transportprotokoll für Multimediadaten die Grundlage für IP-Telefonie dar, sowohl für H.323 als auch für SIP.

3.1. QoS Grundsätze

Hinter all den QoS Überlegungen steht als Ziel eine optimale Nutzung der vorhandenen Ressourcen eines Netzwerkes. In jedem Netzwerk teilen sich die Clients eine gewisse fixe Übertragungsbandbreite. Ist das Netzwerk bis zu 10% ausgelastet, werden QoS Maßnahmen meist überflüssig. Ist das Netzwerk voll ausgelastet, wird auch der beste QoS Mechanismus nicht mehr den gewünschten Effekt bieten können. In diesem Fall hilft nur noch die Verringerung des Traffics oder ein Upgrade der Hardware für mehr Bandbreite. Wofür also QoS?

QoS soll zwei Ziele erfüllen.

Erstens sollte der QoS Mechanismus die vorhandenen Ressourcen möglichst „fair“ unter den Clients aufteilen.

Zweitens soll der per User anfallende Datenverkehr geglättet werden um stoßartige Netzwerkbelastungen (*bursts*) abzufangen und diese in kontinuierliche Datenströme zu verwandeln. Dies führt zu einer höheren Belastbarkeit des gesamten Netzes (vgl. STILLER, 1995, S.4).

Dabei sollte beachtet werden, dass diese Mechanismen nicht zu komplex sind und ihrerseits selbst unnötigen *Overhead* erzeugen. Hier kommt wesentlich der wirtschaftliche Aspekt zu tragen. Werden neue Leitungen benötigt oder kann QoS die vorhandenen Leitungen besser, bzw. effizienter nutzen? Rentiert sich die Investition in QoS, was die Leitungen eventuell zusätzlich belastet? Erschwerend kommt hinzu, dass QoS Mechanismen oft herstellerabhängig sind und die Netzwerk-hardware belasten.

Grundsätzlich werden 2 Arten unterschieden:

1) Integrated Services

2) Differentiated Services

(vgl. TRUMMER, 2001, S.24)

Am einfachsten wäre eine Überdimensionierung des Netzes, womit QoS überflüssig wäre. Wenn genug Bandbreite und schnelle Netzwerk-hardware zur Verfügung steht, sollten auch Echtzeitübertragungen kein Problem darstellen. Allgemein sollte dies für Ethernet gelten, allerdings steigt auch hier der Bandbreitenbedarf ständig an.

3.2. Integrated Services

Dieses Konzept beruht darauf, gewissen Applikationen bzw. Datenflüssen, welche besondere Anforderungen stellen, Vorrang zu geben, ohne den restlichen Datenverkehr über Gebühr zu stören. Ein Beispiel hierfür wäre RSVP, welches *Integrated Sevices* in IP-Netzen ermöglicht.

Um für jede Applikation die geforderten Parameter einzuhalten, müssen die Ressourcen von einem Bandbreitenmanagement kontrolliert und zwischen den Applikationen aufgeteilt werden. Dafür wurden mehrere Serviceklassen definiert, die durch Anwendungstypen unterschieden werden (vgl. PETERSON, 2000, S. 295).

- **Guaranteed Service**: für intolerante Applikationen mit festgelegter maximalen Verzögerungszeit;
- **Controlled Load**: für tolerantere Applikationen, welche zwar eine konstante Verzögerungszeit erwarten, sich aber auf eventuelle Schwankungen einstellen können.

Die Zukunft wird zeigen, ob mehrere Serviceklassen gebraucht werden, da diese beiden nur den Anfang darstellen (vgl. TRUMMER, 2001, S.25).

Die Skalierbarkeit solcher Dienste ist jedoch nur begrenzt möglich: Integrated Services bzw. RSVP bedeuten höheren Rechenaufwand für die Router, da zu jedem vorhandenen Datenfluss eine Ressourcen-reservierung existieren kann, welche zu berücksichtigen wäre. Wenn nun z.B. in einem großen Netz viele Teilnehmer gleichzeitig über VoIP telefonieren (Callcenter etc.) kann das sehr schnell zu unerwartet vielen Datenflüssen führen, welche beim Routing beachtet werden müssen. Um diesem Problem entgegenzuwirken und die zu speichernden Statusinformation pro Router zu reduzieren, wurde unter anderem das Differentiated Service entwickelt, welches in Kapitel 3.3. behandelt wird.

RSVP – Resource Reservation Protocol

Die meisten heutigen Netzwerke inklusive des Internets verwenden das IP Protokoll, welches anders als z.B. ATM verbindungslos nach dem *Best-Effort* Prinzip vermittelt. Um auch hier QoS zu realisieren kann man z.B. das Integrated Services Konzept aufgreifen. Hierbei ist ein zusätzliches Protokoll vonnöten, welches für den gewünschten Pfad im Netzwerk bestimmte Ressourcen reserviert. Allerdings erfordert dies eine Anpassung der Netzwerkinfrastruktur mittels RSVP tauglicher Hardware. Deshalb wurde RSVP von der IETF in der RFC 2205 und RFC 2748 standardisiert.

Voraussetzungen für RSVP

Um eine RSVP gestützte Übertragung zu ermöglichen, müssen zwei Bedingungen erfüllt werden. Als erstes muss vor Beginn der

Übertragung eine Reservierung abgesendet werden, welche die gewünschten QoS - Parameter enthält. Alle beteiligten Router müssen diese Reservierung bestätigen, wodurch ein reibungsloser Ablauf sichergestellt wird. Für bidirektionale Übertragungen muss das doppelt geschehen, einmal für die Hin- und einmal für die Rückroute.

Die zweite Voraussetzung besteht darin, dass alle Router sämtliche stattfindenden Datenübertragungen überwachen und kontrollieren. Dieses sogenannte *Monitoring* bzw. *Traffic Policing* (Transportrichtlinien) sind notwendig, um eventuell auftretende Überschreitungen der ausgemachten QoS - Parameter zu erkennen, und um entsprechend darauf reagieren zu können. Um die in IP-Netzen üblichen stoßartig auftretenden Spitzen abzufangen unterstützen viele Router zusätzlich das sogenannte „*Traffic Shaping*“ (Verkehrsanpassung) um eine kontinuierlichere Lastverteilung zu erreichen (vgl. COMER, 1995, S.549ff).

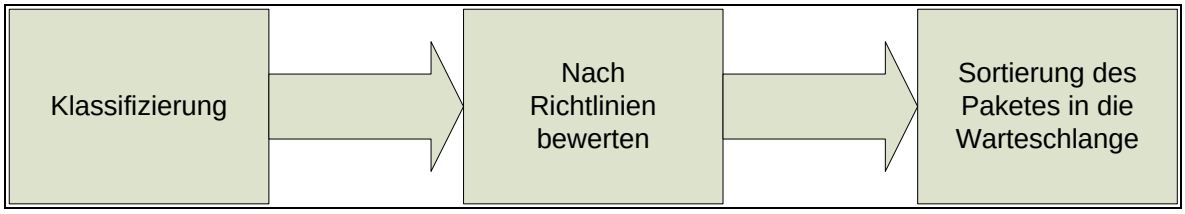

Abbildung 13 RSVP Schema

Um nun das Priorisieren des gewünschten Verkehrs zu ermöglichen, müssen die eingehenden Pakete von einem Traffic Classifier (Klassifizierung) untersucht werden. Anhand dessen werden die Pakete nach den gewünschten Richtlinen bewertet und dementsprechend in die Routerwarteschlangen eingereiht. Dabei ist zu beachten, dass keinerlei Fragmentierung geschieht, da der Classifier die Pakete u.a. anhand der Portnummern der TCP bzw. UDP Pakete erkennt und diese nur im Transportheader enthalten sind, welcher nur im ersten Fragment

definiert ist. Darüber hinaus darf keine Verschlüsselung verwendet werden, welche den *Header* der Transportschicht einschließt.

Funktion RSVP

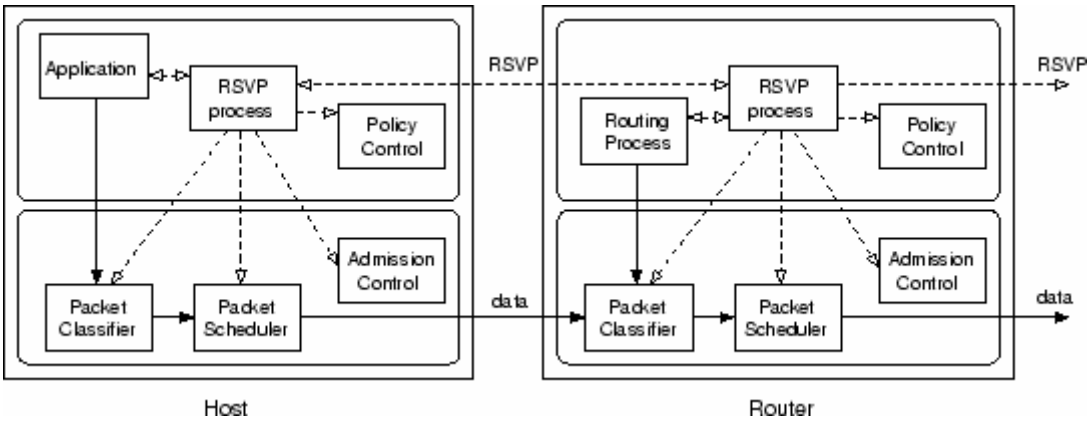

Abbildung 14 RSVP
(nem01.altervista.org/source/Integrated_Services_IntServ.html)

Der Sender schickt eine spezielle Nachricht zum Empfänger, die *RSVP Path Message* Damit wird ein möglicher Pfad vom Sender zum Empfänger ermittelt. Die dabei passierten Router werden protokolliert und so dem Empfänger mitgeteilt.

Entlang des protokollierten Pfades schickt der Empfänger dann eine weitere Nachricht, die RSVP *Reservation Message*. Diese enthält eine sogenannte Flussspezifikation, die die Anforderungen für die Reservierung beschreibt.

Die Router auf dem Weg reservieren die Ressourcen entsprechend dieser Flussspezifikation oder schicken eine Fehlermeldung zurück. Kommt die *RSVP Reservation Message* beim Sender an, kann dieser sich auf die Reservierungen verlassen und gemäß der Spezifikation senden.

Eigenschaften von RSVP

RSVP Reservierungen sind sehr flexibel, und es gibt eine Reihe von Möglichkeiten, wie sie ablaufen können. Neben der Pfadreservierung ist es auch möglich, Sammelreservierungen für mehrere Datenströme (*Shared Reservation*) durchzuführen. Dies ist für *Multicast* Übertragungen sehr sinnvoll, aber auch wenn ein Teilnehmer Datenströme von mehreren Sendern gleichzeitig empfängt. Ein weiteres Feature ist die Wahl zwischen expliziter oder *Wildcard* Spezifizierung der Sender. Der Empfänger definiert somit seine gewünschten QoS - Parameter explizit für einen Sender oder per *Wildcard* für eine Gruppe von Sendern.

Allerdings steigt, wie bereits erwähnt, mit der Zahl der RSVP unterstützten Datenverbindungen auch der *Overhead* gewaltig an. Erschwerend kommt hinzu, dass alle beteiligen Router dieses Protokoll auch richtig interpretieren müssen, was vor allem bei älterer Hardware nicht immer der Fall ist.

3.3. Differentiated Services

Bei den Differentiated Services handelt es sich um ein Verfahren zur Übertragung von Echtzeit-Applikationen über Computernetzwerke. Hier werden die Dienste in wenige QoS-Klassen untersteilt, wobei jeder Dienstklasse ein Satz an Regeln zur Verfügung steht. Bei den Differentiated Services werden zumindest zwei QoS Klassen unterschieden:

- **Best-Effort Klasse**: Bestehender „normaler“ Verkehr
- **Premium Klasse**: Bevorzugter Verkehr

Die Klassifizierung der Pakete erfolgt nach den gewünschten Parametern, danach werden die Pakete entsprechend der Auswertung markiert. Anhand dieser Markierung wird das Paket bevorzugt behandelt oder verworfen.

Zuerst erfolgt die Klassifizierung der Pakete, anhand welcher diese bewertet und anschließend markiert werden. Diese Markierung wird dann in einer Verkehrsanpassung oder dem Verwerfen einzelner Pakete verarbeitet.

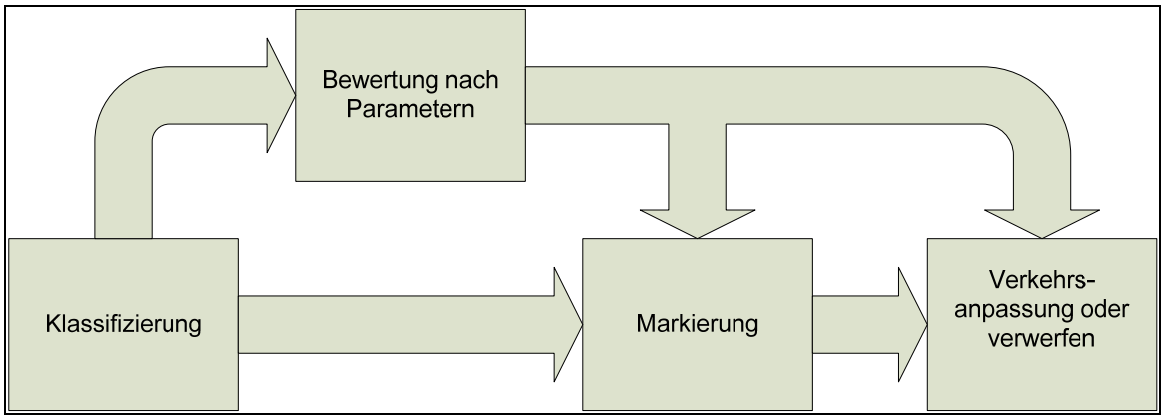

Abbildung 15 Schema Differentiated Services

Bei den Differentiated Services muss nicht jeder Datenfluss verwaltet werden, im Gegensatz zu den Integrated Services. Die Daten werden nur am Eingang des Differentiated Services Netzes bearbeitet und die QoS-Klassen festgelegt. Dazu wird das DSCP-Feld im Differentiated Services Header entsprechend markiert. Dieses Feld entspricht dem Type-of-Service-Feld (TOS) im IPv4-Header beziehungsweise dem Traffic-Class-Octet im IPv6 und dient dazu die einzelnen Datenströme eines Verkehrsbündels zu differenzieren. Das DSCP-Feld hat 6 Bit Länge und kann daher 64 Klassen repräsentieren.

Die zentrale Komponente dieses Konzeptes ist das SLA (Service Level Agreement). Da dieses Konzept einem wirtschaftlichen Nutzen dient, ist das SLA ein Vertrag zwischen dem Kunden und dem Netzbetreiber, in welchem die Datenklasse dem jeweiligen QoS Level zugeordnet wird. So kann für den Kunden wichtiger Verkehr priorisiert werden. Der Provider kann somit seinen Kunden verschiedene Dienstgüten zu verschiedenen Preisen anbieten (vgl. HUTTER, 2000, S.21).

Von der kaufmännischen Seite abgesehen, lässt sich das Differentiated Services Konzept natürlich auch für die Priorisierung von Echtzeitdatenübertragungen im großen Firmennetzen nutzen.

Funktion Differentiated Services

Um die Information der Klasse dem Router mitzuteilen, wird hierbei aber nicht ein eigenes Protokoll wie RSVP bei den Integrated Services verwendet, sondern jedes Paket soll diese Information selber beinhalten. Am einfachsten ist dies über ein Steuerungsbit zu realisieren, welches dem Router mitteilt, ob es „normal“ oder „premium“ ist (vgl. TRUMMER, 2001, S.25f).

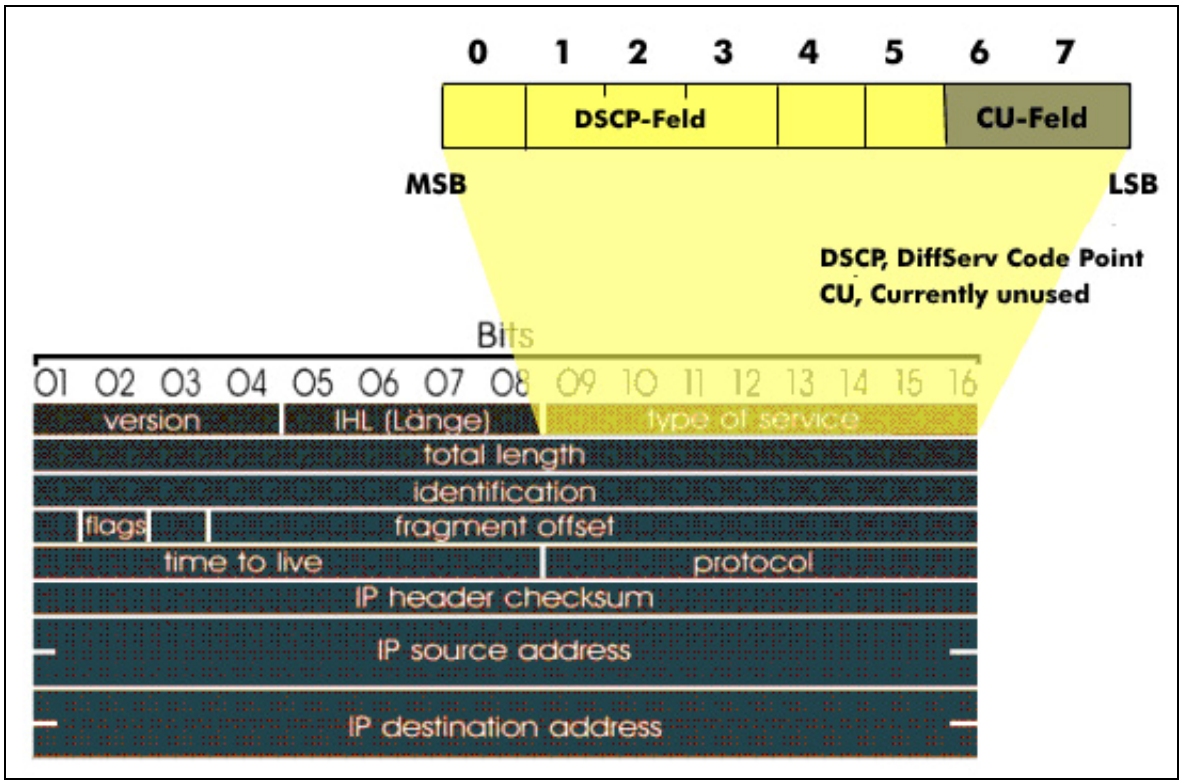

Abbildung 16 Differentiated Services
(networks.siemens.de/solutionprovider/_online_lexikon/7/f011627.htm)

Das Problem hierbei ist, wer dieses Bit setzt und wer sich daran hält. Um dieses Problem zu lösen, spezifizierte die IETF eine Reihe von Verhaltensregeln, die für diese Pakete einzuhalten sind. Diese werden als *Per-Hop Behaviors* bezeichnet. Dafür wurde das *Type of Service* Feld des IP-Headers neu definiert. Da dieses Feld nie Verwendung gefunden hat, wurde es kurzerhand übernommen, um so mehrere Steuerbits für diese QoS Maßnahme zu erhalten. Das TOS (*Type of Service*) Feld im IP-Paket wurde zu diesem Zweck in DSCP (*Differentiated Service Code Point*) umbenannt. (vgl. TRUMMER, 2001, S.25ff).

So soll erreicht werden, dass verschiedene QoS Stufen für verschiedene Datenströme möglich werden, ohne zusätzlichen Signalisierungs-aufwand wie bei den Integrated Services in Kauf nehmen zu müssen. Der Sender kann somit ohne vorherige Vorabadministration mit dem Senden seiner Daten beginnen. Bei Netzen, die über einen oder

mehrere Providern verbunden werden, ist auch darauf zu achten, dass die DCSP Bytes SLA konform gesetzt werden, da sonst die Router des Providers nicht wissen, was priorisiert werden soll, bzw. dies falsch interpretieren.

Neben dem Sender oder einem Client kann auch ein Router, den ein Paket durchläuft, dieses Feld setzen (z.B. die Gateways des Providers). Dabei werden mehrere Verhaltensmuster, PHB (*Per Hop Behavior*) festgelegt, etwa die *Drop Threshold* (Toleranzschwelle für einen Paketdrop), Pufferzuordnung oder Priorität und Datenrate eines Dienstes. Diese Muster werden auf den DSCP (*Differentiated Services Code Point*) abgebildet, wobei sich einem PHB mehrere DSCPs zuteilen lassen.

Vorausetzung sind allerdings Router, welche *queue scheduling* (Warteschlangensteuerung), bzw. *queue management* (Warteschlangenmanagement) unterstützen, da sie sonst die QoS Klassen der Pakete nicht unterscheiden können.

Die Funktionsweise von Differentiated Services ist relativ einfach. Der erste Router (auch *Ingress Router* genannt) kennzeichnet das DSCP - Feld. Anschließend passt er das Paket an, klassifiziert und überprüft es. Die restlichen Router passen dann ihre Queues diesen Paketen an, sofern sie dies unterstützen.

Im Internet werden *Differentiated Services Domains* definiert, welche für bestimmte Netzabschnitte gelten. Die IETF definiert eine solche Domain als Abschnitt wo dieselben SLA’s gelten, welche unter gemeinsamer Verwaltung stehen. Solche Domains können ein autonomes System, bestimmte geographische Netzregionen oder ein Netzwerkabschnitt mit bestimmter Netztechnik sein. Zwischen mehreren Domains existieren dann *boundary components* (Rand-komponenten), welche zwischen den Domains Verbindungen herstellen.

Es gibt zwei Arten dieser *Components*, die *Ingress Nodes* für eingehenden Verkehr und die *Ergress Nodes* für ausgehenden Verkehr. Diese passen die DSCP Bits den jeweiligen SLA's an. Um dies zu ermöglichen, kennen diese *Nodes* die jeweiligen SLA's der angrenzenden Domains. Diese Anpassungen an das jeweilige SLA werden im TCA (*Traffic Conditioning Agreement*) definiert (vgl. HUTTER, 2000, S.23f).

3.4. QoS integrierende Netzwerke

Als QoS integrierende Netzwerke bezeichnet man Netzwerke, welche vom grundlegenden Design bereits auf die Implementation von QoS konzipiert sind. Als Beispiel dient der ATM (*Asynchronos Transfer Mode*) Mode. Dieser erlaubt, genauso wie IP basierende Netzwerke, den Transfer von einzelnen Paketen und die Verwaltung von mehreren logischen Verbindungen über eine physikalische Schnittstelle. Der wesentliche Unterschied liegt allerdings in der fix vorgegeben Größe der Pakete von 53 Bytes. Diese werden als Zellen bezeichnet und bestehen jeweils aus 5 Bytes *Header* und 48 Bytes Nutzdaten (vgl. STALLINGS, 1997, S. 327ff). Ein weiterer wesentlicher Unterschied ist, dass IP verbindungslos arbeitet. Erst mit Hilfe darüberliegender Protokolle wie z.B. TCP kann bei IP eine Verbindung hergestellt werden. Im Gegenzug dazu ist bei ATM der Verbindungsaufbau mit dem Kommunikationspartner erforderlich. Hierbei werden auch gleich die QoS-Parameter festgelegt. Auch hier existieren verschiedene Serviceklassen (vgl. TRUMMER, 2001, S. 26ff).

- **Non-Realtime -** Variable Bitrate: Die Spezifikation dieser Bitrate liegt in Form eines Token Bucket vor. Darunter versteht man eine durchschnittliche Transferrate, welche durch eine Mittelwert-bildung von kurzen stoßartigen Datentransfers entsteht.

- **Realtime -** Variable Bitrate: Wie Non-Realtime, allerdings Angabe der maximalen Verzögerungszeit.
- **Constant Bitrate**: Wie *Realtime* nur mit kontinuierlichem Datenstrom.
- **Unspecified Bitrate**: Für den „normalen“ Verkehr, entspricht dem *best-effort* bei IP.
- **Available Bitrate**: Mit dynamischer Anpassung der Bitrate an die aktuelle Belastungssituation des Netzwerkes.

4. Netzwerk Simulatoren

4.1 Überblick Netzwerksimulatoren

4.1.1. Opnet

Opnet ist ein diskret ereignisgesteuerter Simulator für Netzwerke. Die Firma Opnet kooperiert mit diversen Universitäten und stellt den Simulator für Bildungseinrichtungen kostenlos zur Verfügung. Der Simulator ist modular aufgebaut und stellt mit offenen Schnittstellen diverse Möglichkeiten zur Verfügung um eigene Scripte miteinzubauen. Die sehr große Anzahl von Nutzern und Entwicklern bietet Hilfe in diversen Foren und Newsgroups. Dieser Simulator unterstützt auch die Simulation von diversen QoS Methoden. Im Gegensatz zu dem NS-2 Simulator, welcher im Kapitel 4.2. näher beschrieben wird, arbeitet Opnet grafisch, was die Eingewöhnung deutlich erleichtert.

4.1.2. AdventNet Simulation Toolkit 5

Der Simulator der Firma AdventNet ist ein sehr komplexer Simulator um komplette Netzwerke zu designen. Wie bei den meisten kommerziellen Simulatoren wird auf einer aufwendigen grafischen Oberfläche jedes Device einzeln (oder per proprietärem Script) konfiguriert. Die Einstellungen der Devices sind den realen Geräten exakt nachempfunden, um diese direkt übernehmen zu können. Dabei sind diverse Graphiken und Informationen sehr schnell per Klick angezeigt. Für die Simulation von QoS eignet sich dieser Simulator nicht unbedingt, da diese Funktionen nicht dezidiert einstellbar sind, bzw. nicht in der Dokumentation angegeben werden.

4.1.3. QualNet Developer

Dieser Simulator der Firma scalable-networks ist ein leistungsfähiges Tool um große Netzwerkszenarien zu simulieren. Wie bei AdventNet wird reale Netzwerkhardware simuliert. Die Konfiguration ist ebenfalls der realen Hardware angepasst. Dies macht allgemeine Aussagen über QoS schwierig, da man mehre Produkte und deren teils proprietären Lösungen testen müsste. Ferner ist dieser Simulator auch für Unix-Plattformen erhältlich.

4.1.4. Weitere Simulatoren für Netzwerke

TM Netsim OMNet++; OPNET; Real 5.0; SSFNet – Scalable Simulation Framework; GLASS – GMPLS Lightwave Agile Switching Simulator; X-Sim; BRITE; CNet; Flan Network Simulator; NAB – Network in a box; JNS – Java Network Simulator; NePSing - Network Protocol Simulator next generation; NCTUNS-2.0 – Network Simulator and Emulator; GloMoSim; jFirewall Simulator; Lanforge Fire & Ice; Unix network configuration simulator; [Blutch] Network Simulator; NetSim Ip/Tcp/Digital Network Simulator; WiNe2; WIPsim – Wireless IP Simulator; CNNA Network Simulator 4.0.

4.2. Der Network Simulator NS-2

Einer der meist verbreiteten Simulatoren ist der NS-2 Simulator. Der NS-2 wurde für diese Arbeit gewählt, da er sich zur Simulation von Integrated- als auch Differentiated Services eignet. Die meisten anderen Simulatoren sind nur für gewisse Protokolle oder Mechanismen gedacht. Die Variabilität und seine freie Verfügbarkeit waren weiters ausschlaggebend für die Verwendung des NS-2 für diese Arbeit.

Ein wichtiger Aspekt war auch die herstellerunabhängige Auslegung des Simulators. Während bei den meisten kommerziellen Produkten ein Router immer ein bestimmtes Modell eines Herstellers ist, arbeitet der NS-2 mit einfachen Nodes. Diese simulieren Router ihrer Definition nach. Man kann die Warteschlangen und die Routingmechanismen beliebig definieren. Des Weiteren wird der NS-2 in einschlägiger Literatur häufig als Standardwerkzeug angegeben.

Der Network Simulator NS-2 ist ein diskret ereignisgesteuerter Simulator für Netzwerktopologien. Er ist frei verfügbar und *open source*, weshalb er zumeist in der Forschung eingesetzt wird. NS-2 unterstützt die Simulation von TCP, Routing und Multicast Protokollen über verdrahtete Verbindungen, aber auch kabellose Verbindungen wie WLAN (*Wireless Local Area Network*) oder Satellit (vgl. MOSIG, 2004, S.3).

4.2.1. Geschichte

Entwickelt wurde der Simulator 1989 an der UC Berkeley als Variante des REAL Network Simulators und sollte eigentlich für Simulationen von dynamischen Aspekten in paketorientierten Netzen wie Lastanalysen

und Staukontrolle dienen. Mit Unterstützung der DARPA und einigen bekannten Instituten wie das Xerox Palo Alto Research Center und das Information Science Institut of Southern California entstand unter dem Namen VINT Projekt 1995 der NS-1, welcher die Skalierbarkeit und Interaktion zwischen den Protokollen untersuchen konnte. 1997 wurde die zweite Version NS-2 fertig gestellt, welche weiter überarbeitet und um neue Simulationsmöglichkeiten, wie RTP, Scheduling und mobile Hosts erweitert wurde (vgl. MOSIG, 2004, S.3).

4.2.2. Eigenschaften

Der Simulator wird ausschließlich über Scriptdateien gesteuert. Er bedient sich einer objektorientierten Struktur mittels C++ und OTcl. Dies ermöglicht es relativ, einfach ein Script zu schreiben, da die Syntax sich an moderne Programmiersprachen anlehnt und somit leicht zu erlernen ist.

Die eigentliche Simulation findet als interne Berechnung statt, das Ergebnis wird als Tracedatei präsentiert, welche man mittels des Network Animators *NAM*, grafisch darstellen kann.

Durch seine Funktion des Realtime-Schedulings ist der Simulator auch fähig, als Emulator zu arbeiten. Somit ist die Erstellung einer virtuellen Maschine möglich, in der echter Datenverkehr zu Echtzeitbedingungen durch eine Netzwerktopologie fließt (vgl. Fall & Varadhan, 2005, S.27ff).

Der Simulator ist in C++ geschrieben und kann dank offener Sourcecodes und unzähliger Dokumentationen und Tutorien durch eigene Zusatzmodule erweitert, oder je nach Bedarf verändert werden.

4.2.3. Anforderungen

Benötigt wird ein PC und ein C++ Compiler. Der Simulator wurde auf FreeBSD entwickelt weshalb er auf nahezu allen UNIX Derivaten wie Linux, Solaris, etc. funktionieren sollte. NS-2 funktioniert mit Hilfe eines Unixemulators auch auf Windows.

4.2.4. Komponenten

Der Simulator besteht aus mehreren modularen Komponenten. Diese Bauweise hat Vorteile, allerdings kann sie auch zu Problemen führen.

- **Tcl/Tk**: Die *Tool Command Language* und das zugehörige grafische Toolkit bilden das Fundament des Systems. In dieser Scriptsprache werden die Steuerscripte geschrieben.
- **OTcl**: Mit dem *Object Tcl* wird das *Tcl* um objektorientierte Fähigkeiten erweitert.
- **TclCL**: *Tcl with Classes* stellt die Verbindung zwischen der Scriptdatei und den in C++ realisierten Objekten her.
- **NS-2**: Das eigentliche Hauptprogram, liest die Steuerdatei und berechnet die Simulation und speichert das Ergebnis in eine Tracedatei. Dies geschieht textbasierend.
- **Nam**: Der *Network Animator* liest die Tracedatei und stellt die Ergebnisse grafisch dar. Dabei bietet er Steuerungs- und Animationsmöglichkeiten. Darüber hinaus enthält er auch einen Editor um eine grafische Netzwerktopologie zu erstellen, welche als *Tcl* Script wieder dem NS-2 zugeführt werden kann.
- **Xgraph**: Ein simpler Plotter um Netzauslastungen als Graph darzustellen, liest ebenfalls Tracedaten.

- **Perl**: Ist eine standardmäßig in Linuxsystemen enthaltene Scriptsprache, welche zur automatischen Erzeugung von Scriptdatein für den Simulator verwendet werden kann.

(vgl. MOSIG, 2004, S.5)

4.2.5. Funktion

Theoretisch kann der NS-2 jeden Netzwerkverkehr simulieren. Durch die Tatsache, dass man eigene Komponenten miteinbauen kann, gibt es im Internet diverse Plugins, auch für ausgefallene Protokolle oder Szenarien.

- **Anwendungsschicht**: Für die OSI Schicht sind u.a. HTTP, FTP, sowie diverse andere TCP Applikationen wie telnet usw. implementiert.
- **Transportprotokolle**: Unterstützt werden TCP, UDP, RTP und diverse Erweiterungen davon.
- **Routing**: Es werden sowohl statische Routen als auch dynamische Routen ermöglicht. Durch die Unterstützung von *distant vector* und *link state* werden RIP (*Routing Information Protocol*) und OSPF (*Open Shortest Path First*) Simulationen möglich.
- **Router Mechanismen**: Bei den Routingmechanismen werden alle gängigen Warteschleifenalgorithmen unterstützt. FIFO (*First In First Out*)/*DropTail*, *Fair Queueing*, *Stochastic Fair Queueing*, *Deficit Round Robin*, *Class Based Queueing* und *Random Early Queueing*.

- **Sicherungsschicht**: Es werden sowohl CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) für Ethernet, als auch CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) für Funknetze unterstützt.

(vgl. MOSIG, 2004, S.6ff)

4.2.6. Bedienung

Gesteuert wird der Simulator mittels einer Scriptdatei. In ihr werden die Netzwerktopologie, die verwendeten Protokolle und Routing-mechanismen deklariert. Des Weiteren teilt man dem Simulator mit, welche Ereignisse während der Simulation auftreten sollen. Das Script wird in folgender Reihenfolge erstellt:

1) Zuerst werden die *nodes* platziert

2) Je zwei Knoten werden verlinkt

3) Jeder Knoten wird mit einem Agenten versehen

4) Den Agenten wird *traffic* hinzugefügt

5) Sender- und Empfänger-Agent werden miteinander verbunden

6) Für bestimmte Zeitpunkte werden *events* gesetzt

4.2.7. Programmierung

Das Script hat immer denselben Aufbau. Dieses Template kann wieder verwendet und erweitert werden. Es folgt ein Beispiel (nach MOSIG, 2004, S.9).

```
# erzeuge eine neue Instanz des Simulators
set ns [new Simulator]

# Legt den Routingmechanismus fest
# default -> static routing
# DV -> distant vector
# LS -> link state
$ns rtproto DV

# Öffnet das nam Tracefile für schreibenden Zugriff
set nf [open output.nam w]
$ns namtrace-all $nf

# Hier stehen die eigenen Kommandos zur Beschreibung
# der Topologie und der Agenten

# Definiert die finish-Funktion, die nach Beenden
# der Simulation nam mit dem tracefile aufruft
proc finish {} {
global ns nf
$ns flush-trace
#Close the trace file
close $nf
#Execute nam on the trace file
exec nam output.nam &
exit 0
}

# Ruft nach 5 Sekunden Simulationszeit die finish-Funktion auf
$ns at 5.0 "finish"

# startet die Simulation
$ns run
```

Nodes

Diese Knoten repräsentieren die einzelnen Computer, Server oder Router. Jedem Node kann ein Name, eine Farbe und eine Position in der Topologie zugeteilt werden.

```
set n1 [$ns node]
```

Links

Zwischen den Nodes werden Verbindungen definiert. Hierzu werden die Parameter eingestellt, die den Link charakterisieren. Dazu gehört

simplex/*duplex*, Bandbreite, Ausbreitungsgeschwindigkeit und eine Warteschlange am Ende.

```
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
```

Agents:

Anschließend wird den einzelnen Knoten jeweils ein Agent zugeordnet. Dieser definiert das verwendete Protokoll. Zu beachten ist, dass serverseitig das Protokoll spezifiziert wird und clientseitig ein entsprechender Abfluss eingestellt wird.

Für UDP ist dies der sogenannte Null-Agent, der alle UDP Pakete verwirft. Bei TCP ist dies der TCPSink-Agent, der noch ein Acknowledge zurücksendet bevor das Paket verworfen wird, da bei TCP alle Pakete bestätigt werden müssen (siehe Kapitel 1.3. TCP).

```
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n2 $null
```

Traffic Generator:

Um Datenverkehr zu simulieren muss auf der Serverseite dem Agenten ein Trafficgenerator zugeordnet werden, welcher Daten an den Client sendet. Der Generator kann konstante Bitraten (CBR) simulieren, aber auch steigende Bitraten. FTP und Telnet sind für die TCP Simulation möglich. Zusätzlich werden noch die Paketgrößen und die Zeitintervalle zwischen den Paketen definiert.

```
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 500
$cbr set interval_ 0.005
$cbr attach-agent $udp
```

Connect Agents:

Am Ende müssen die einzelnen Agents miteinander verbunden werden, damit jeder Quelle auch eine Senke zugeteilt wird.

```
$ns connect $udp $null
```

Events:

Um diverse Situationen simulieren zu können, werden nun Ereignisse festgelegt, welche zu einem bestimmten Zeitpunkt auftreten. So ein Event wäre z.B. das Ausfallen einer Leitung oder das Starten der Trafficgeneratoren.

```
$ns at 0.5 "$cbr start"
$ns rtmodel-at 1.0 down $n1 $n2
$ns rtmodel-at 1.5 up $n1 $n2
$ns at 2.0 "$cbr stop"
```

4.3. Test des NS-2 Simulators

Um die Funktion des Simulators zu verifizieren, wird ein einfacher Versuchsaufbau simuliert und im Labor experimentell nachgestellt. Das Ziel ist es, mit denselben Parametern, gleiche Ergebnisse sowohl bei dem Laborexperiment und als auch bei der Simulation zu erreichen.

Zwei Linuxrechner werden mit zwei Cisco Routern der 2500 Baureihe verbunden. Diese sind durch eine 2 Mbit Serial Verbindung verbunden.

Ein Rechner erzeugt einen TCP und einen UDP Stream, während der andere diesen aufzeichnet. Dazu wird der D-ITG (*Distributed Internet Traffic Generator*) verwendet, welcher auf der Senderseite den gewünschten Verkehr generiert und gleichzeitig auf der Empfängerseite diesen in einer Datei aufzeichnet.

Um die Ergebnisse besser vergleichen zu können, werden die aufgezeichneten Daten bei beiden Versuchen gleich dargestellt.

Hardware:

- Cisco 2520 Router (IOS 12.2(12a))
- Cisco 2503 Router (IOS 12.2(12a))

Software:

- Ubuntu Linux – Version 2.6.14-2-686
- D-ITG, Distributed Internet Traffic Generator

Verbindungsdaten:

- TCP (FTP): Packetsize 1500 bytes
- UDP (CBR): Packetsize 1000 bytes; rate 140 kbyte/s

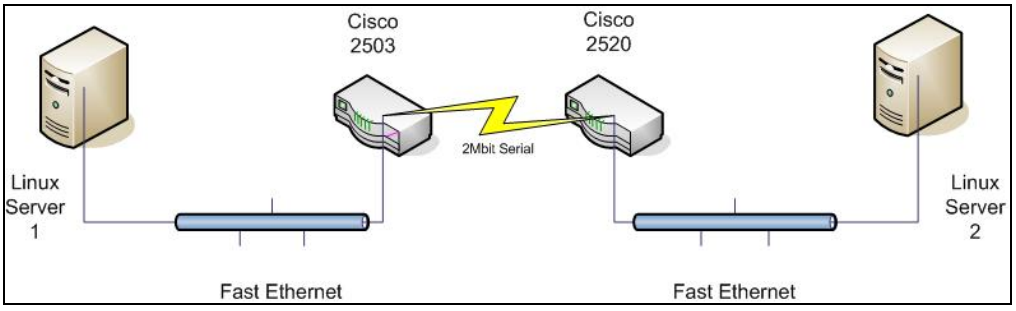

Abbildung 17 Simulatortest

Ergebnis der NS-2 Simulation:

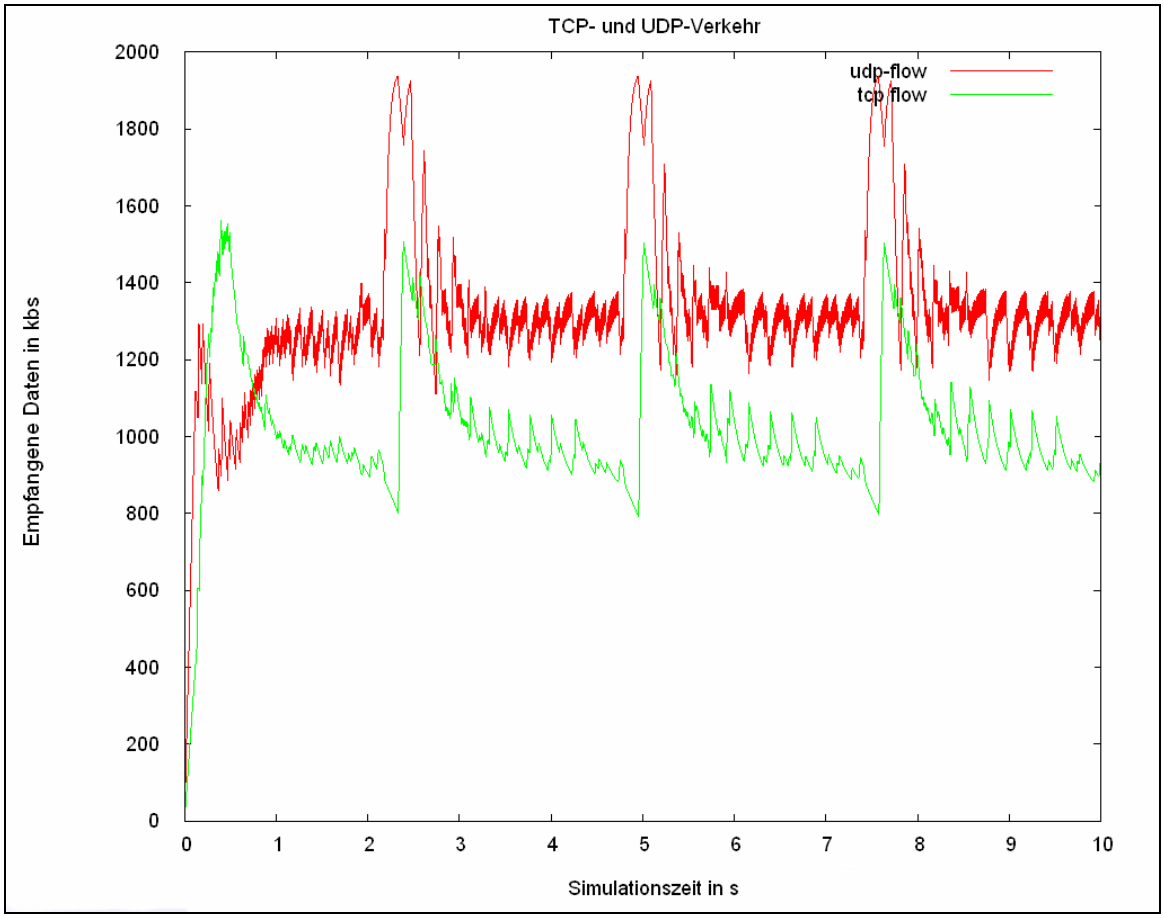

Abbildung 18 Network Simulator

Bei der Simulation ist das wechselseitige Spiel zwischen TCP und UDP um die begrenzt vorhandene Bandbreite gut ersichtlich. In diesen mittels EWMA (*Exponential Weighted Moving Average*) aufbereiteten Grafiken ist der TCP und UDP Stream ähnlich, da der UDP Stream auf ca. 1,1 Mbit (140 Kbyte) eingestellt wurde. Ansonsten würde er den TCP Stream verdrängen. Diese Werte sind absichtlich so gewählt, um das Wechselspiel zwischen TCP und UDP besser zu verdeutlichen.

Ergebnis des Experimentes:

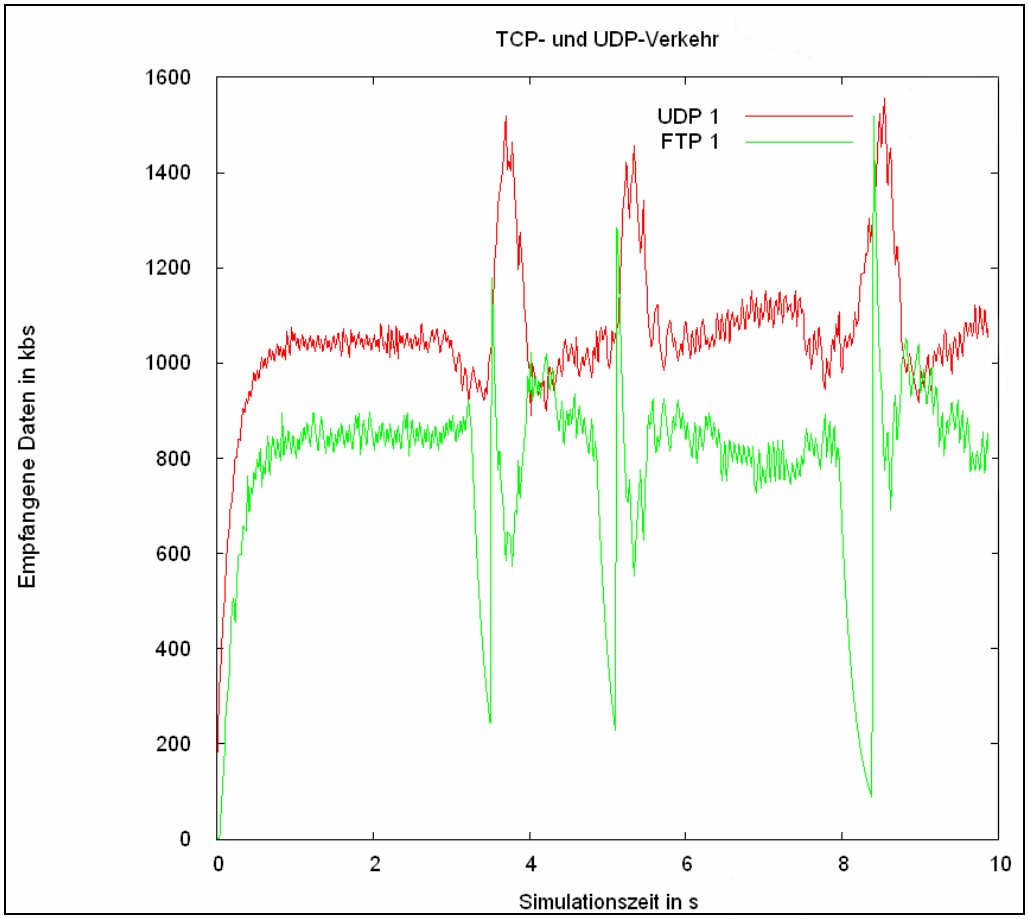

Abbildung 19 Laborexperiment

Die Abbildung 19 zeigt das Resultat des Laborversuches mittels der zwei Router und Server. Wie schon bei der Simulation ersichtlich, versucht UDP die maximal verfügbare Bandbreite zu bekommen, während sich TCP aufgrund seiner Acknowledgements anzupassen versucht. Durch die voreingestellten fixen Bitraten bei TCP steigert sich das *Sliding Window* am Anfang bis es sich konstant auf 22 Pakete (bzw. 32.767 Bytes) einpegelt, was das Maximum des Linux-Kernels darstellt. Das lässt sich dadurch erklären, dass die TCP Senderseite unablässig Pakete sendet, ohne erst einmal auf die Bestätigung der Übertragung der Pakete zu warten. Wenn der Empfangspuffer von 32.767 Bytes aufgefüllt ist, kann es dazu kommen, dass ein oder mehrere Paket

fehlen um das *Window* zu vervollständigen. Danach wartet der Empfänger auf die nochmalige Übertragung des fehlenden Paketes, welches er mehrmalig vom Sender nachfordert. Dadurch stockt die gesamte TCP Übertragung kurzfristig, da der Empfänger seinen Puffer erst verschieben kann, wenn er alle fehlenden Pakete erhalten hat. Erst dann kann er seinen Puffer auf einen Schlag leeren, um weitere Pakete annehmen zu können. Aufgrund dieser Tatsache kommt es bei Sekunde 3.5; 5 und 8.5 zu kurzfristigen Einbrüchen in der TCP Übertragung.

In dem Moment wo der TCP Verkehr stockt wird Bandbreite frei, welche der UDP Verkehr sofort nutzt, was dessen Ausschlag kurz nach den TCP Einbrüchen erklärt.

Die geringfügig höhere Bandbreite des Simulators resultiert aus dem bei dem Experiment nicht beachteten Protokolloverheads. Ansonsten sind die Resultate ohne Probleme miteinander vergleichbar, was die richtige Funktion des Simulators bestätigt.

5. QoS Simulation

Das Ziel der Simulation ist es, den Einfluss der QoS Mechanismen auf die Übertragungsqualität des VoIP Verkehrs darzustellen.

5.1. Installation des Simulators

Der Simulator wurde für Unix und Linux Umgebungen geschrieben. Er ist frei unter der URL http://www.isi.edu/nsnam/ns/ erhältlich. Es gibt ein Install-Script, welches den Simulator automatisch installiert. Für diese Arbeit wurde der Cygwin Unix Emulator für Windows verwendet. Für diesen existieren eigene Patchfiles um den Simulator zu installieren. Für die Simulation wird die Version 2.26 verwendet, da das RSVP Plugin diese voraussetzt.

5.2. Versuchsaufbau

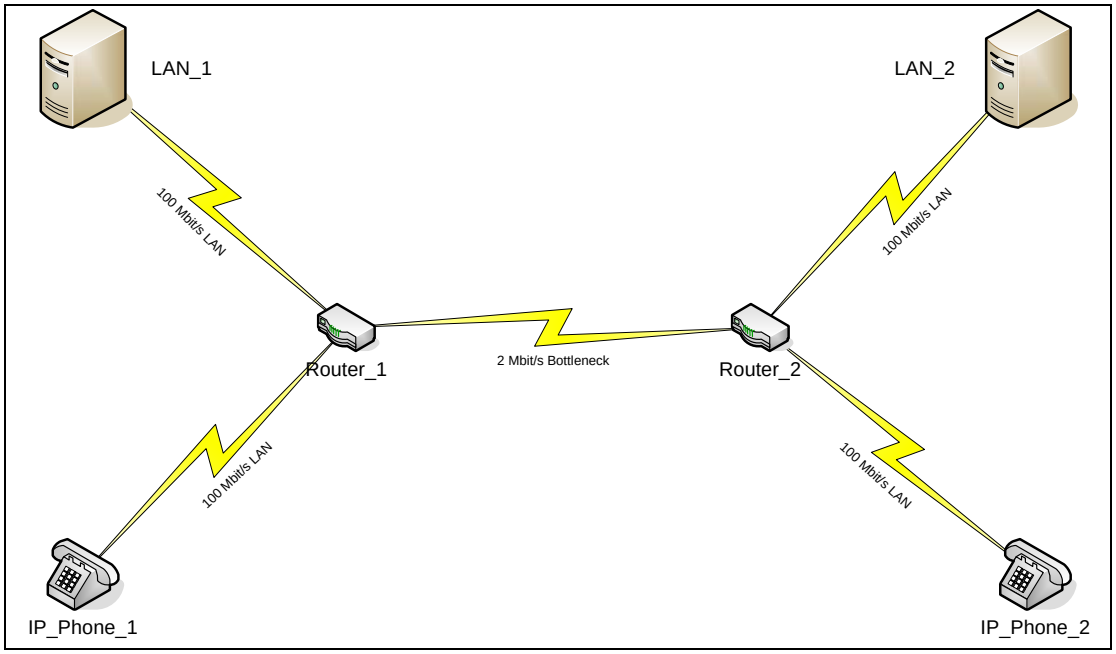

Abbildung 20 Versuchsaufbau

Um die Vorteile von QoS deutlich sichtbar machen zu können, wird ein Netzwerkengpass simuliert. Dies stellt eine Firma mit zwei Standorten dar, welche über eine Standleitung verbunden sind. Um nicht riesige Datenmengen generieren zu müssen, ist der Engpass mit 2 Mbit/s und 10 ms *delay* relativ gering gewählt, lässt aber gute Rückschlüsse auf andere Engpässe und mehr Verkehr zu. Die restlichen Verbindungen sind mit 100 Mbit/s und 2 ms *delay* realisiert.

Um den normalen Netzwerkverkehr zu simulieren, kommunizieren zwei Kommunikationsendpunkte über den Engpass miteinander. Diese simulieren normalen TCP bzw. UDP Verkehr, wie er in jedem LAN auftritt. Somit simulieren diese zwei Punkte, „LAN_1&2“ genannt, das Firmennetzwerk.

Daneben wird eine RTP Übertragung mit den typischen VoIP Parametern aufgebaut, um ein Gespräch über den Engpass zu simulieren.

Um die QoS Methoden bewerten zu können, wird die Übertragung hinsichtlich sechs verschiedener Graphen bewertet, welche für die Simulationen mit und ohne QoS gleich sind, um direkte Vergleiche zu ermöglichen.

5.3. Simulation: Annahmen

Um vergleichbare Aussagen treffen zu können wurde bei bei allen Simulationen derselbe, genau definierte Verkehr simuliert. Die Parameter sind wie folgt:

- Simulationszeit: 20 Sekunden
- Queuelänge (Router_1, Router_2): 25 Pakete je Richtung

VoIP Parameter

Das VoIP Telefonat wird zwischen zwei simulierten Endpunkten aufgebaut. Diese Verbindung besteht aus zwei RTP Übertragungen inklusive des RTP Protokoll *Overhead.* Dabei teilt es sich einen Engpass mit den jeweils zwei TCP und UDP Übertragungen.

- 8 kbit/s G.729a CS-ACELP Codec (Paketgröße 10 bytes, dazu kommt der RTP-Header mit 12 bytes *Overhead* (dies simuliert der NS-2 automatisch; vgl. ITU, 2005))
- Verbindung 1: zwischen IP_Phone_1 und IP_Phone_2 via RTP (voice1) – startet bei Sekunde 0,1
- Verbindung 2: zwischen IP_Phone_1 und IP_Phone_2 via RTP (voice2) – startet bei Sekunde 0,1

TCP Parameter

Der TCP Verkehr wird mittels zwei versetzt startenden FTP Datenströmen simuliert. Dabei ist die Paketgröße so groß wie möglich gewählt. Um die unterschiedlichen Auswirkungen von TCP und UDP auf den VoIP (UDP) Verkehr sichtbar zu machen, läuft der TCP Verkehr nur in eine Richtung, also stört er nur den voice1 Verkehr.

- Paketgröße: 1500 bytes
- Verbindung 1: zwischen LAN_1 und LAN_2 via TCP (traffic1) – startet bei Sekunde 0,1
- Verbindung 2: zwischen LAN_1 und LAN_2 via TCP (traffic2) – startet bei Sekunde 2

UDP Parameter

Der UDP Verkehr wird mit Hilfe von zwei versetzt startenden CBR (*constant bit rate*) Verbindungen simuliert. Diese werden mit einer sehr hohen Datenrate ausgesendet, um den Engpass zu überlasten. Die Datenrate entspricht in etwa einem Videostream. Wie bei TCP wird auch der UDP Verkehr nur in eine Richtung versandt, allerdings in die Gegenrichtung, um den voice2 Verkehr zu stören.

- Paketgröße: 1000 bytes
- Datenrate: 1,5 Mbit/s
- Verbindung zwischen LAN_2 und LAN_1 via UDP (cbr1) – startet bei Sekunde 1
- Verbindung zwischen LAN_2 und LAN_1 via UDP (cbr2) – startet bei Sekunde 3

5.3. Simulation ohne QoS

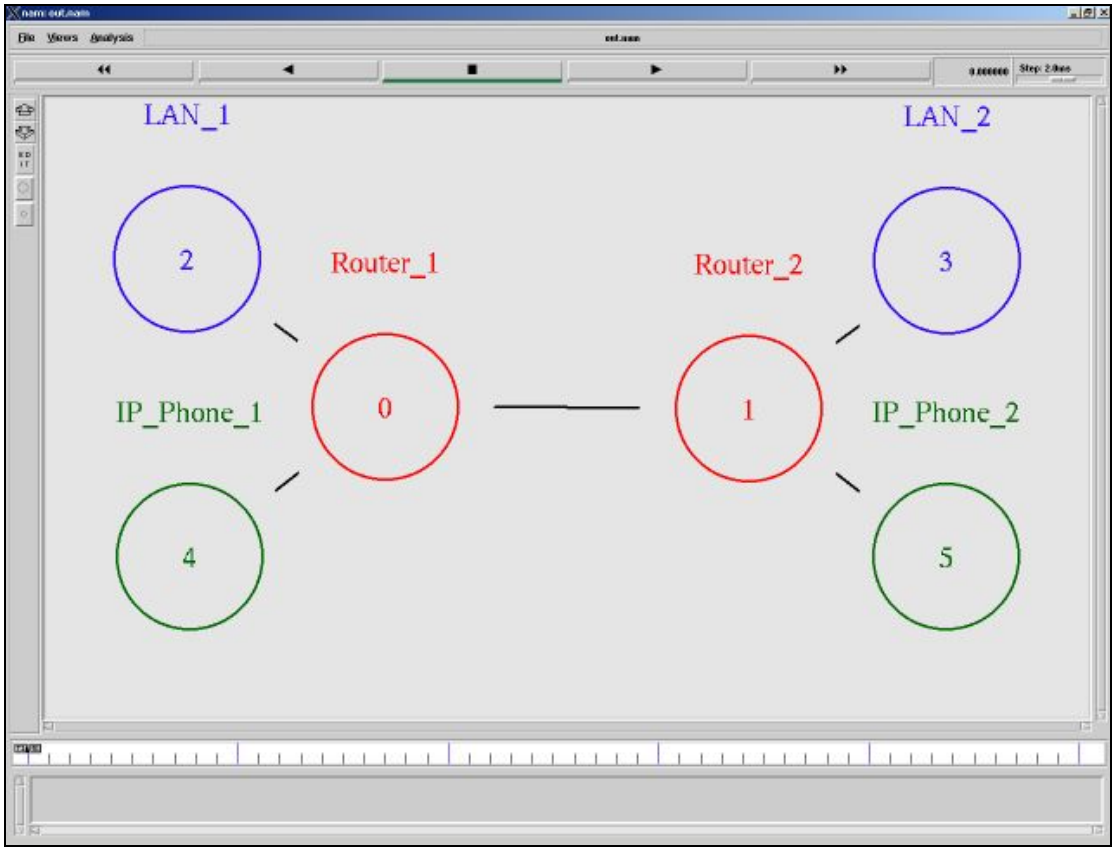

Abbildung 21 NAM Darstellung

Die Abbildung 21 zeigt das grafische Fenster des NAM. Hier ist der Versuchsaufbau im NS-2 realisiert. Am oberen Bildrand ist die Simulationsteuerung sichtbar, am unteren Bildrand die Zeitlinie. Um den Verkehr zwischen den einzelnen Komponenten darzustellen kann der NAM den Paketfluss grafisch darstellen wie in Abbildung 22 ersichtlich.

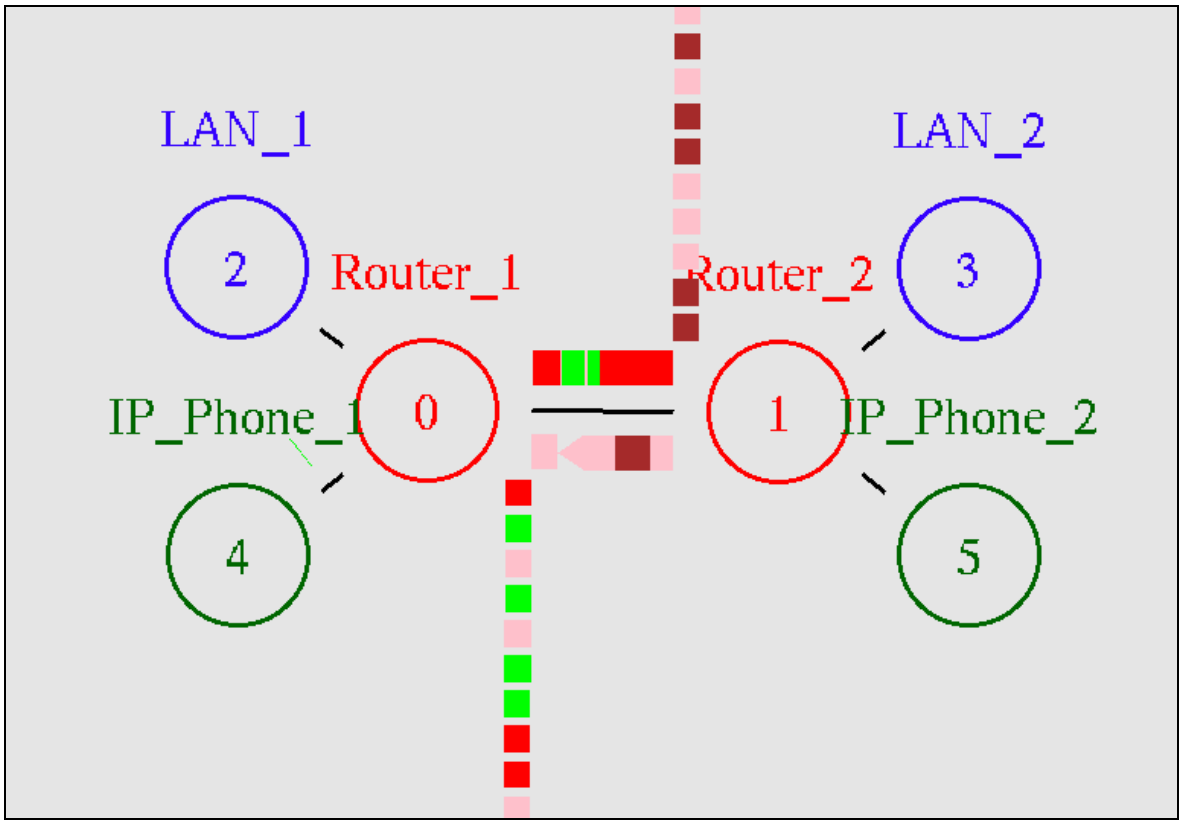

Abbildung 22 NAM Simulation

Unter Router_1, bzw. über Router_2 stellt der NAM die Queues der jeweiligen Router grafisch dar. Die verschiedenen Farben symbolisieren verschiedene Arten von Paketen, in diesem Fall sind zwischen LAN_1 und LAN_2 zwei FTP Transfers, dargestellt in rosa und blau. Dazu kommen noch auf derselben Strecke UDP Transfers (rot und orange). Der VoIP Verkehr via RTP (UDP, incl. Signalisierung) ist zwischen dem IP-Phone_1 und IP_Phone_2 in grün und braun gehalten.

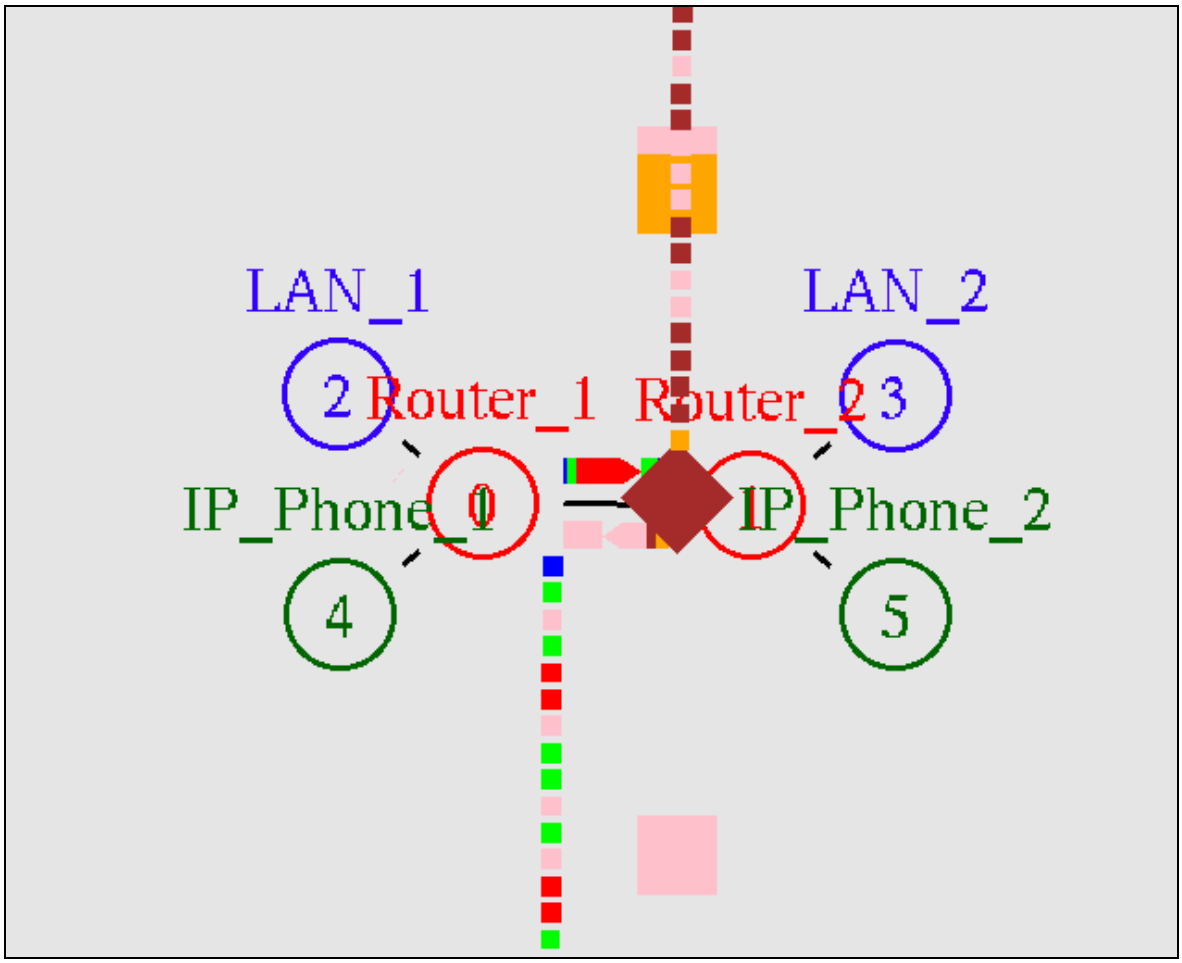

Abbildung 23 NAM Simulation Packet Loss

In Abbildung 23 ist die grafische Darstellung eines Paketverlustes zu sehen, da auf Router_2 die Queue überlastet ist. Man sieht hier gut den Verlust sowohl von UDP als auch von RTP und TCP Paketen (in Form von großen, herunterfallenden Quadraten). Da hier noch kein QoS implementiert ist, behandelt der Router die Pakete gleich. Diese Grafiken dienen allerdings nur zum leichteren Verständnis und zur Kontrolle, da sie wenig aussagekräftig sind.

5.3.1. Voice (RTP) Verkehr

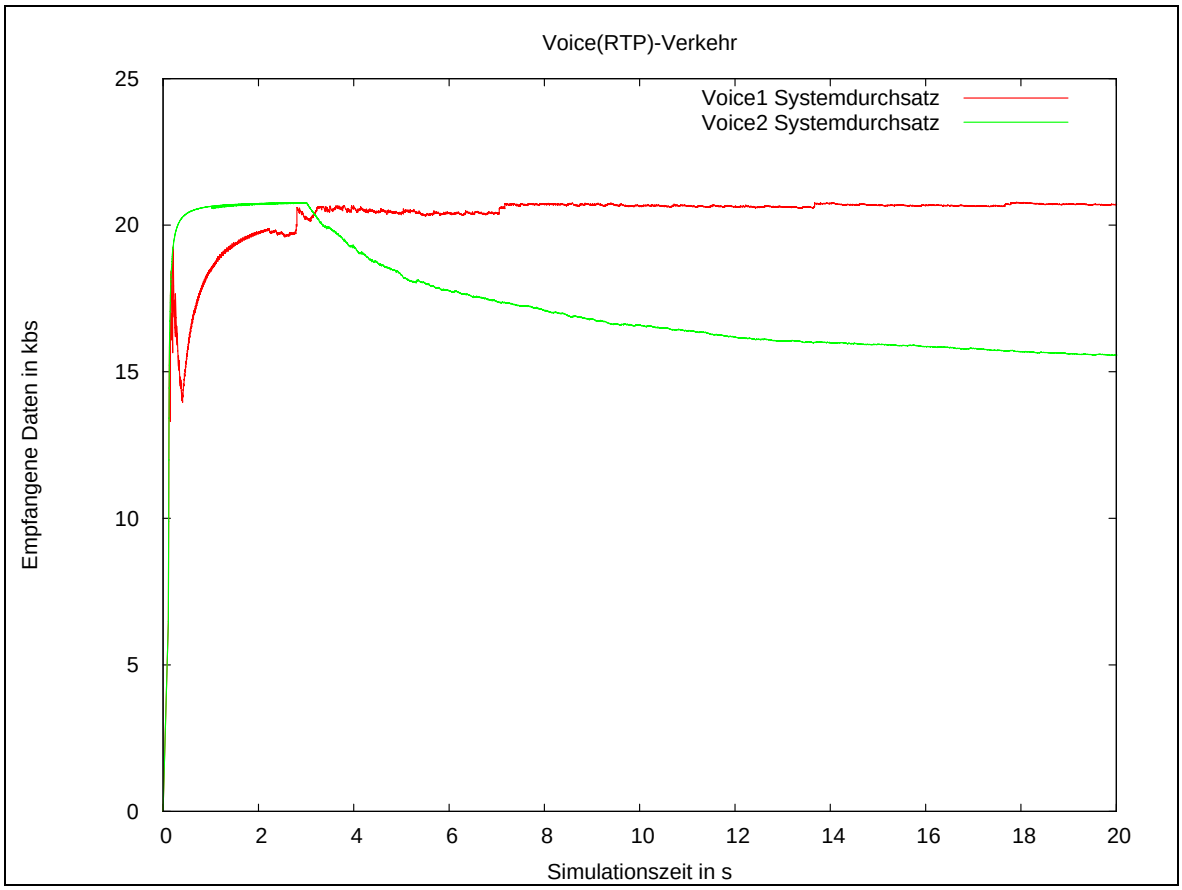

Abbildung 24 Voice (RTP) Verkehr normal

Die Voice (RTP) Verkehrssimulation zeigt die von den RTP Paketen benötigte Bandbreite auf dem Hauptstrang (zwischen Router_1 und Router_2). Diese durchschnittliche Bandbreite wird mittels EWMA (*Exponential Weighted Moving Average*) errechnet. Sie pendelt sich bei 20 kbit/s ein, was genau der Bandbreite von 8 kbit/s des RTP und des RTP *Overhead* von 12 kbit/s entspricht.

Bei dem voice1 Stream (rot) wird das gleichzeitige Starten des FTP Verkehrs durch einen Einbruch ab Sekunde 0,1 sichtbar.

Der Grund dafür ist, dass die 1500 Bytes großen TCP Pakete eine gewisse Zeit benötigen, um versandt zu werden. Dadurch werden die

RTP Pakete in der Router Wartschlange gereiht und werden bis zu 6 ms (die Dauer eines Transfers eines TCP Paketes) pro gereihtem TCP Paket verzögert.

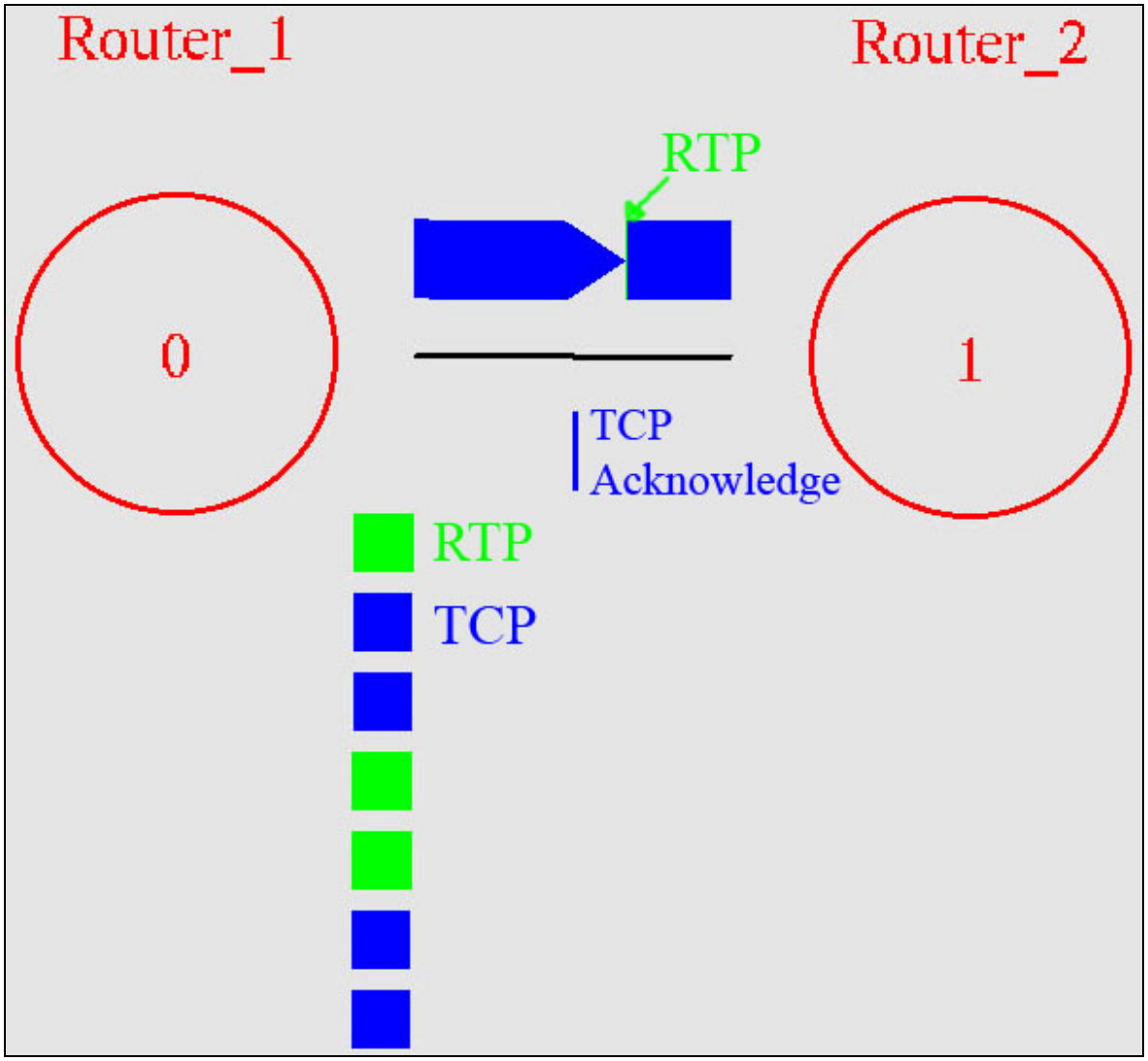

Abbildung 25 Voice (RTP) Verkehr Detail

Die Grafik 25 zeigt die Routerwarteschlange des Router_1, wo die kleinen (grünen) RTP Pakete zwischen den großen (blauen) TCP Paketen eingereiht werden. Bei der Übertragung zwischen den zwei Routern ist auch ein einzelnes RTP Paket zwischen zwei TCP Paketen sichtbar.

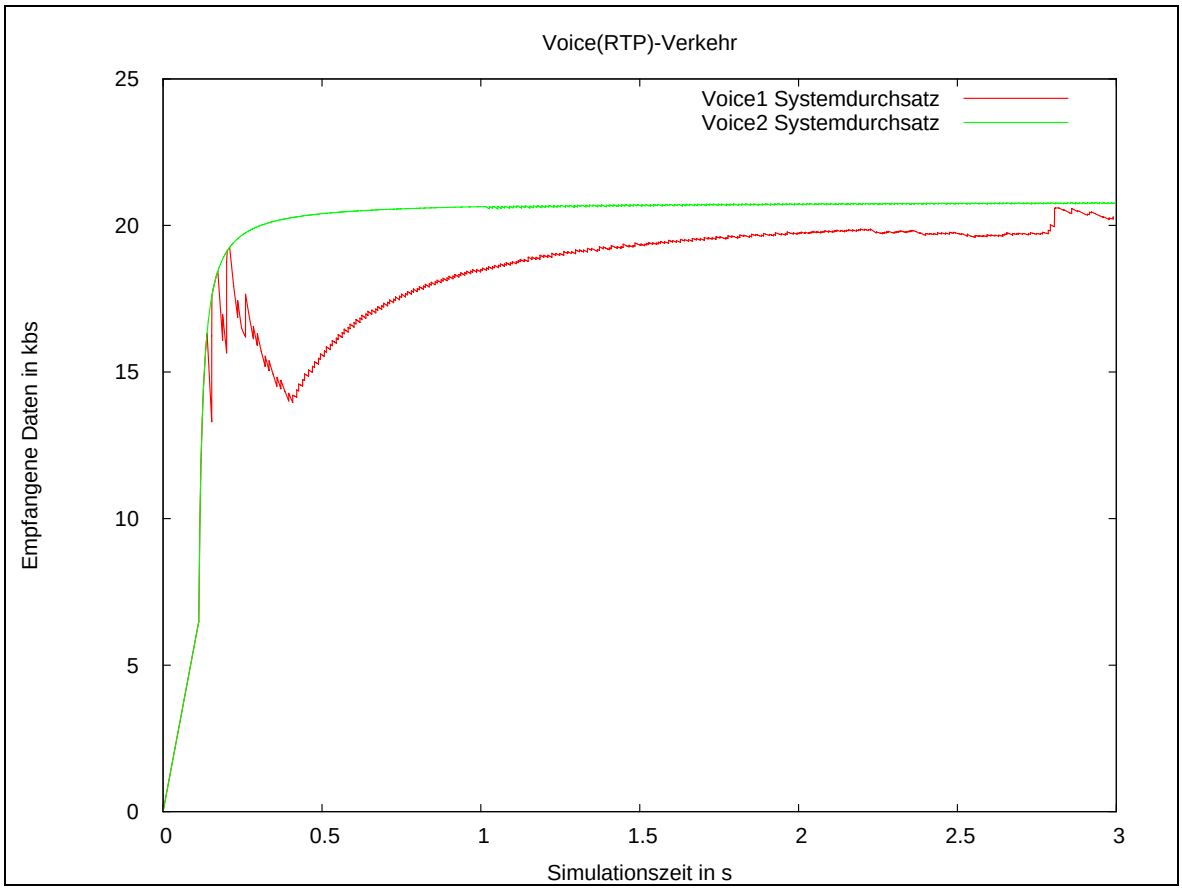

Abbildung 26 Voice (RTP) Verkehr Detail 2

Wie in der Detailgrafik sichtbar, entspricht der Einbruch ca. 8 kbit/s (bzw. mindestens einem Paket).

Der Verkehrsfluss pendelt sich danach automatisch ein, da nach einer gewissen TCP Synchronisationsdauer die Anzahl der TCP Pakete an die Leitung angepasst werden und die RTP Pakete gleichmässig in die RTP Transferlücken verschränkt werden.

Der grüne Graph (Voice2) ist bis zum Start des zweiten UDP Verkehrs (Sekunde 3) stabil, fällt danach wegen der konkurrierenden UDP Pakete ab, da diese ohne Acknowledgements versendet werden, und die Warteschlange im Router auslasten. Dadurch kommt es zum Paketverlust wegen Überfüllung und Qualitätseinbußen bei der VoIP Übertragung. Der beschränkende Parameter ist die Bandbreite des

Kanals mit 2 Mbit/s, die bereits maximal von den etablierten Strömen ausgenutzt wird, wodurch der Router nicht mehr alle anstehenden Pakete verarbeiten und versenden kann.

5.3.2. TCP und UDP Verkehr

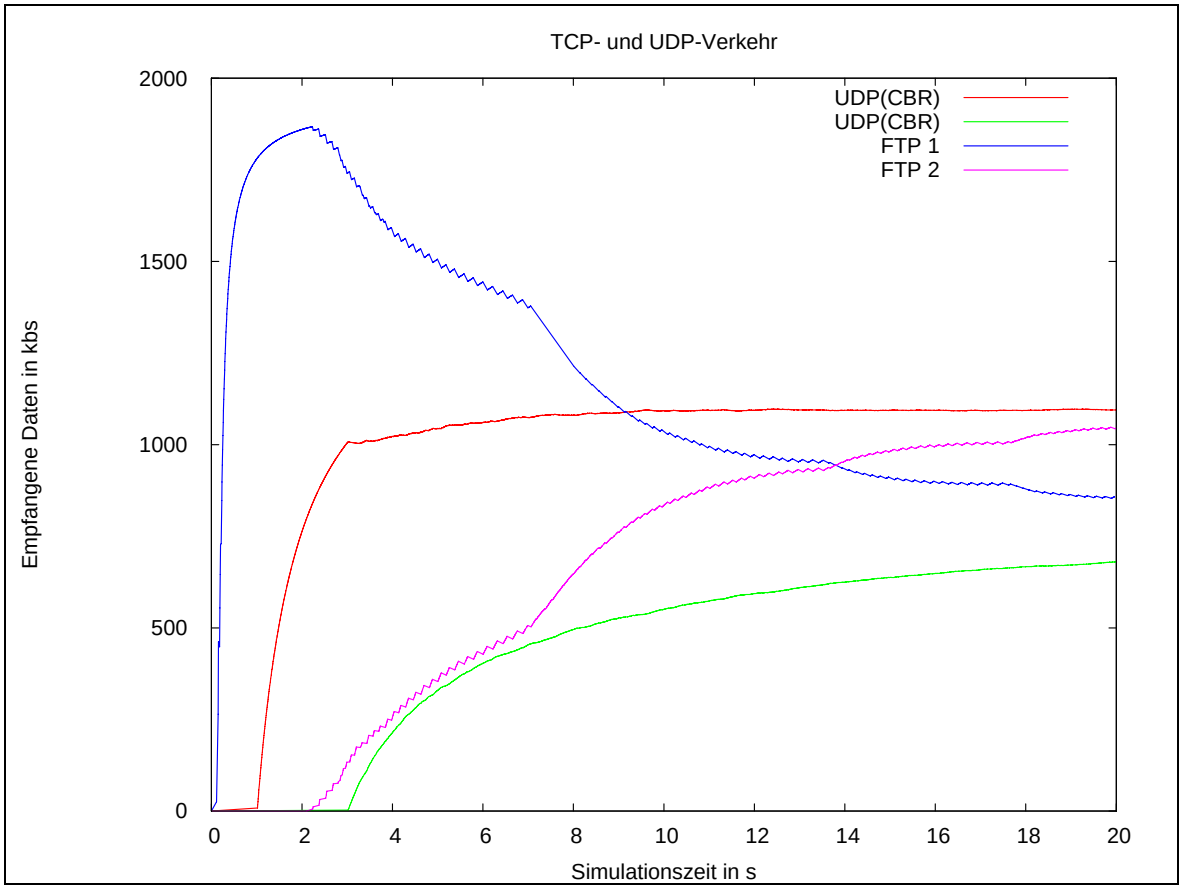

Abbildung 27 TCP und UDP Verkehr normal

Die TCP- und UDP- Grafik zeigt die durchschnittliche Bandbreite auf dem Hauptstrang zwischen Router_1 und Router_2, wobei der TCP Verkehr von LAN_1 nach LAN_2 fließt und der UDP Verkehr in umgekehrter Richtung von LAN_2 nach LAN_1 verläuft. Es steht so in jeder Richtung 2 Mbit/s zur Verfügung.

Der erste TCP Stream wird durch den bei Sekunde 2 einsetzenden zweiten TCP Stream gebremst, da sie sich die verfügbare Bandbreite teilen müssen und beide auf die ankommenden Acknowledgements warten müssen, wodurch der *Sliding Window* Mechanismus zum Tragen kommt, welcher eine Durchsatzbegrenzung (effektive Bandbreite) der beiden Ströme zur Folge hat. Somit teilen sich die beiden TCP Ströme die verfügbare Bandbreite annähernd gleich auf.

Gut ersichtlich ist, dass der erste TCP Datenstrom (FTP 1) seinen Durchsatz immer wieder korrigieren muss, in diesem Fall bedingt durch den neu hinzugefügten zweiten TCP Datenstrom, da dieser auch um die verfügbare Bandbreite konkurriert. Der zweite FTP Datenstrom baut am Anfang protokollmässig eine Verbindung auf und beginnt zu senden. Bedingt durch den Acknowledgement Mechanismus, wird die anfängliche Übertragungsrate des Erstsendenden herabgesetzt, da die Pakete des neu Hinzugekommenden in die Warteschlange gereiht werden, und dadurch die zurückgesendeten Acknowledgements verzögert werden, was in einer Reduzierung des effektiven Paket-durchsatzes resultiert. Im Endeffekt werden beide Ströme so verschränkt, dass sie annähernd gleich im Router verarbeitet werden und sich dementsprechend ihre Durchsatzrate angleicht.

Ungeachtet der Tatsache, dass der TCP Stream nur in eine Richtung und der UDP Stream in die entgegengesetzte Richtung läuft, kommt erschwerend hinzu, dass die UDP Streams den TCP Stream dadurch beeinflussen, dass die TCP Acknowledgements den Rückweg über den mit UDP Verkehr belasteten Weg nehmen müssen und in die strapazierte Routerwarteschlange eingereiht werden.

Auf der Seite der beiden UDP Streams wird ersichtlich, dass die beiden Ströme zu senden beginnen und somit die gesamte verfügbare Bandbreite ausnutzen.

5.3.3. Voice Delay

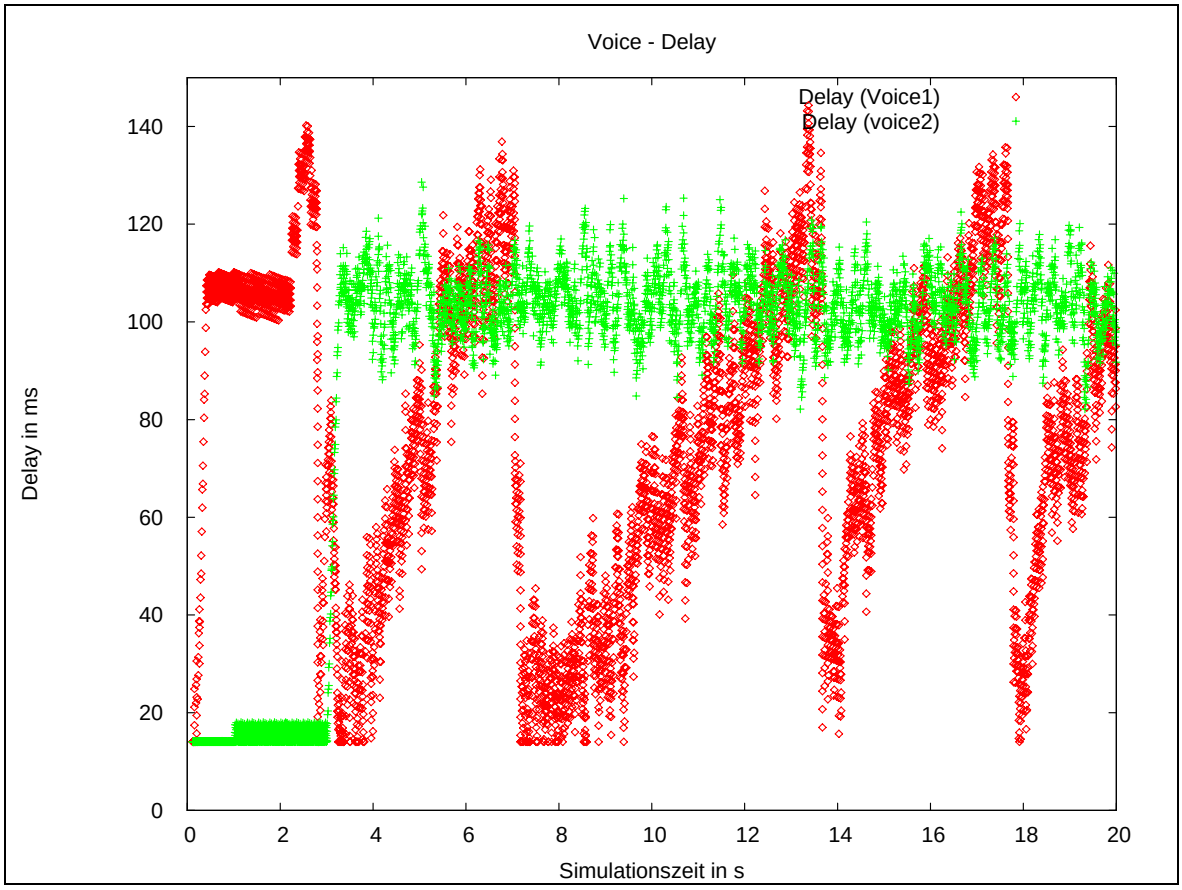

Abbildung 28 Voice Delay normal

Die Voice Delay Simulation zeigt die Übertragungsdauer der einzelnen RTP Pakete zwischen dem IP_Phone_1 und dem IP_Phone_2 in ms.

Die mindestens benötigte Zeit von IP_Phone_1 zu IP_Phone_2 beträgt 14 ms, bedingt durch die in den Simulationsparametern festgelegten Leitungsverzögerungen von 10 ms zwischen den Routern und jeweils 2 ms von den Routern zu den Telefonen.

Bei dem durch TCP gestörten RTP Verkehr zwischen IP_Phone_1 und IP_Phone_2 wird gleich am Anfang ersichtlich, dass die RTP Pakete durch die steigenden Anzahl der TCP Pakete in den Routerwarteschlangen immer stärker verzögert werden. Jedes TCP

Paket benötigt ca. 6 ms zum Versand.

Die starken Verringerungen der Verzögerung bei Sekunde 3, 7, 13 und 18 resultieren aus einer überfüllten Warteschlange bei Router 2, wodurch kurzzeitig die TCP Acknoledgements verzögert oder auch verworfen werden, weshalb der TCP Verkehr momentan gestoppt wird und die RTP Pakete so sofort auf Router 1 durchgereicht werden. Dies wird auch im „TCP und UDP Verkehr normal“ Diagramm (Abb. 23) durch kurze Einbrüche sichtbar.

Bei der durch UDP gestörten Verbindung von IP_Phone_2 zu IP_Phone_1 tritt ab Sekunde 3 durch den einsetzenden zweiten UDP Stream eine dauerhafte, hohe Störung auf. Diese resultiert aus der mit vielen UDP Paketen überlasteten Routerwarteschlange des zweiten Routers. Dadurch werden die RTP Pakete stark verzögert, oder, wie im Diagramm „gedroppte Pakete normal Voice“ (Abb. 27) zu erkennen ist, verworfen. Davor tritt kaum Verzögerung auf, da der erste UDP Stream alleine die Leitung nicht voll auslastet.

Für sehr gute Sprachübertragung ist ein Delay von unter 150 ms erforderlich. Für den Gesamtdelay muss man für die Encodierung und Decodierung der Sprachcodecs und den Jitterpuffer noch zusätzliche Verzögerungen hinzurechnen.

5.3.4. Voice Jitter

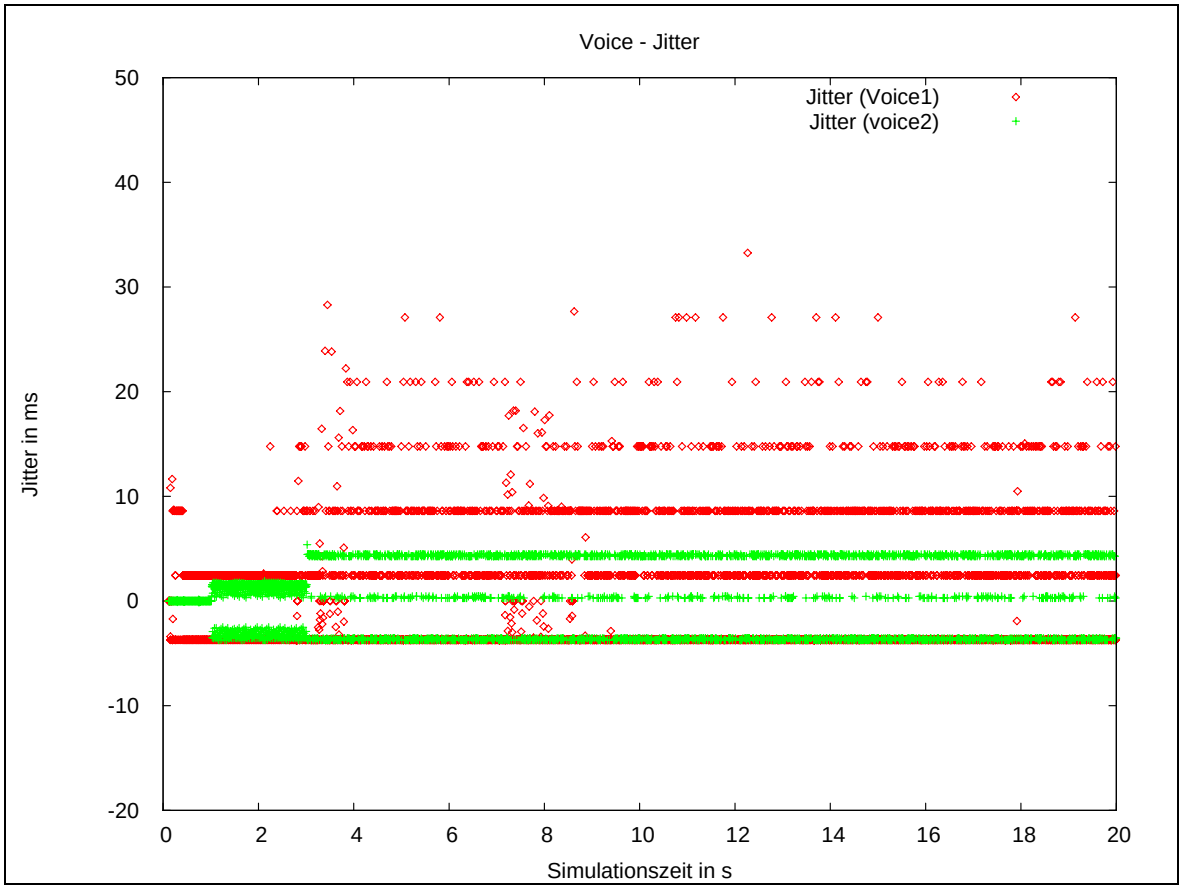

Abbildung 29 Voice Jitter normal

Die Voice – Jitter Grafik zeigt das unterschiedliche Einlangen der RTP Pakete an den IP_Phones_1 und IP_Phone_2. Diese Laufzeit-unterschiede werden als Jitter bezeichnet. Um eine angemessene Sprachqualiät zu gewährleisten übernimmt ein Jitterpuffer die sequenzielle und chronologische richtige Reihung der Pakete (siehe Kap. 2.1.2). Je geringer die Streuung des Jitters ausfällt, desto geringer muss der Jitterpuffer bezüglich seiner Pufferzeit dimensioniert sein.

Der lokalen Häufungen des ersten Jittergraphen (Voice1) treten genau im Abstand von 6 ms auf. Dies rührt von der Tatsache her, dass ein 1500 Byte großes TCP Paket genau 6 ms benötigt, um über den Hauptstrang verschickt zu werden (1500 Bytes / 2 Mbit/s) und die RTP

Pakete zwischen diese Pakete gereiht werden. Wie oben beschrieben tritt auch hier hauptsächlich bei Sekunde 3, 7 und vereinzelt bei Sekunde 18 eine lokale Disturbanz auf.

Bei dem zweiten Jittergraphen (Voice2) treten die lokalen Häufungen im Abstand von 4 ms auf, was der Versanddauer eines 1000Bytes großen UDP Pakete entspricht (1000Bytes / 2Mbit/s).

Interessant ist, dass der TCP Verkehr augenscheinlich störender auf den Jitter wirkt als der UDP Verkehr, wobei aber außer Acht gelassen wird, dass ab Sekunde 3 ca. 25% der RTP Pakete verworfen werden und so gar nicht mehr den Graphen einfliessen.

5.3.5. Queuelängen

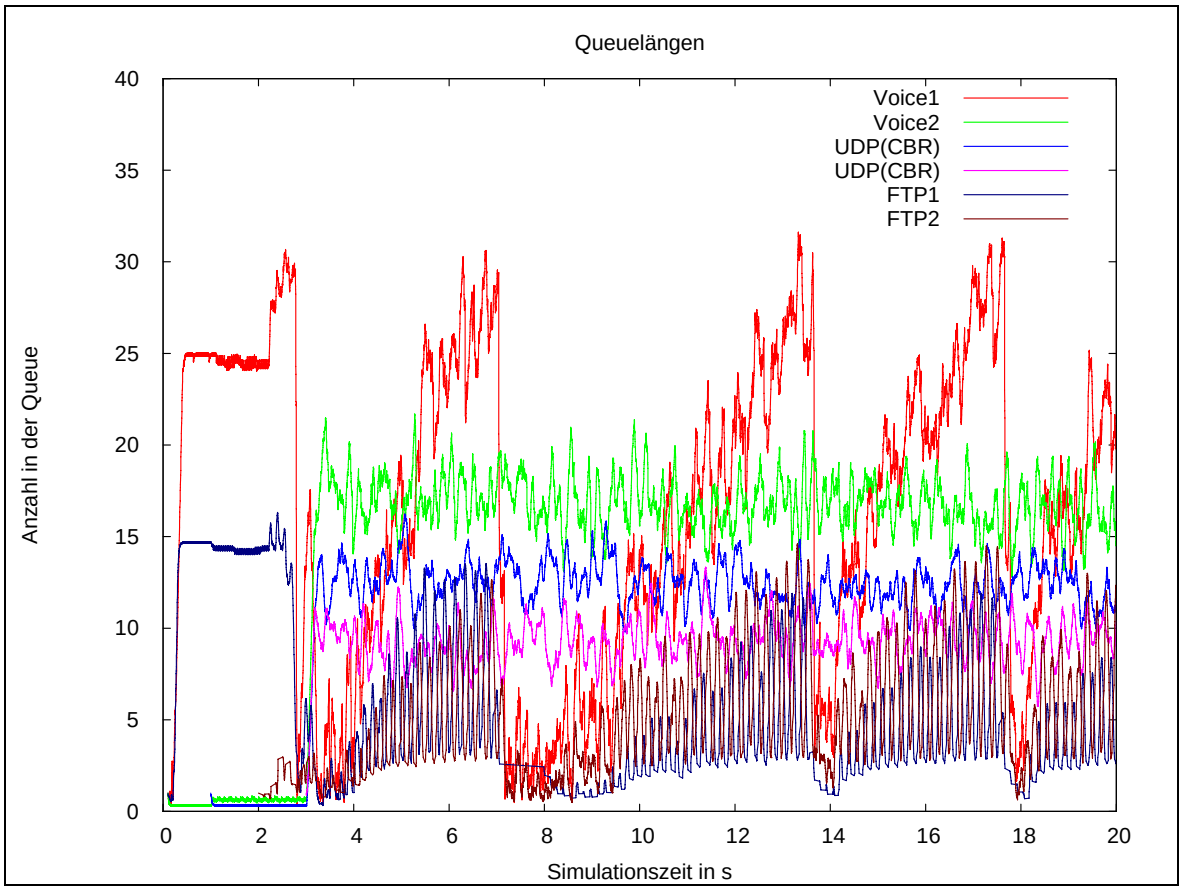

Abbildung 30 Queuelängen normal

Die Queuelängensimulation zeigt die momentan vorhandenen Pakete in den Queues der beiden Router (Router_1 und Router_2).

Die Gesamtqueuelänge beträgt 50 Pakete. Da die RTP Pakete sehr klein sind (10 bytes), müssen auch relativ viele gesendet werden um eine Transferrate von 8 Kbit/s zu erreichen, was sich in einer hohen Anzahl an RTP Paketen in den Routerwarteschlangen bemerkbar macht. Anhand dieses Graphen kann man die durch die Warteschlangen-situation verursachten Beeinträchtigungen im Verkehr-, Jitter- und Delaygraphen verstehen.

5.3.6. Gedroppte Pakete

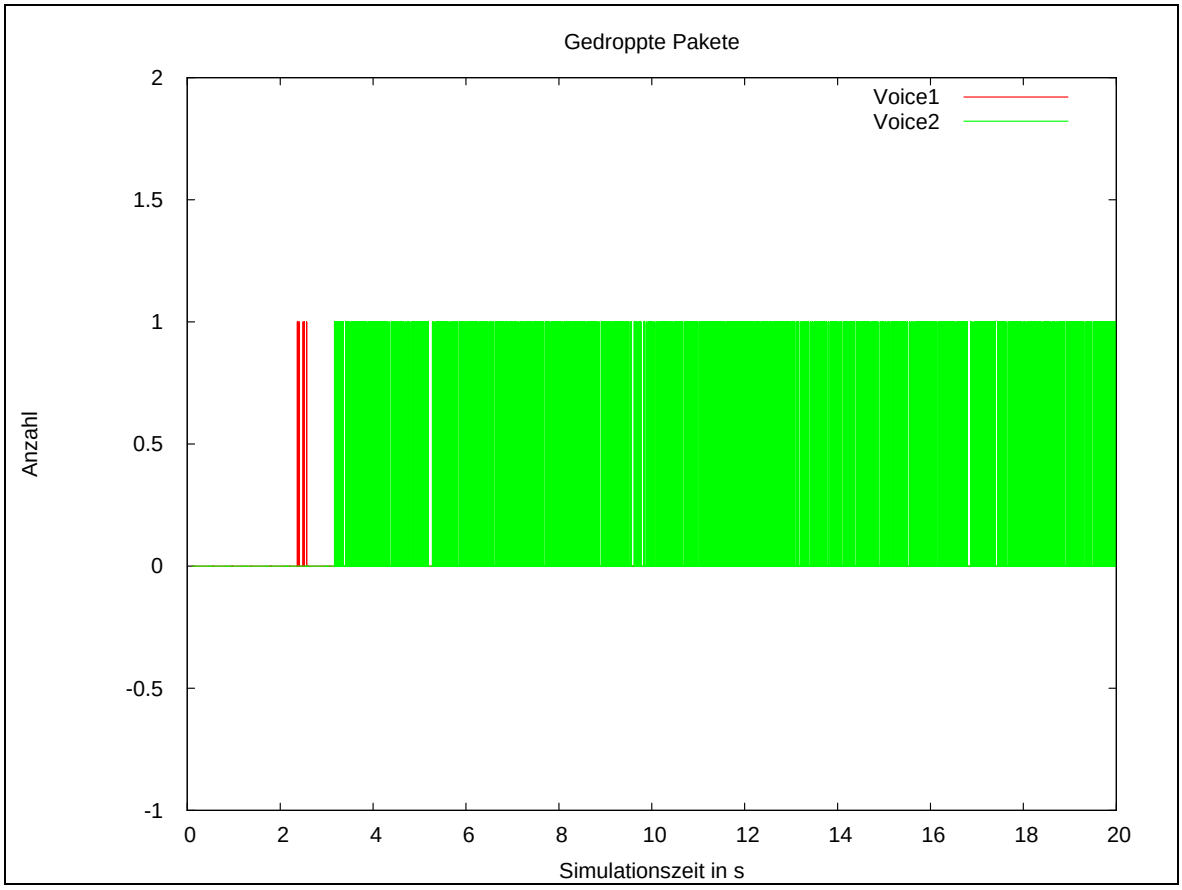

Abbildung 31 Gedroppte Pakete normal Voice

Die Grafik der gedroppten Pakete zeigt, wann ein RTP Paket von einem der beiden Router (Router_1 und Router_2) aufgrund überlasteter Warteschlangen verworfen wird. Jede Linie bedeutet ein gedropptes Paket.

Da RTP Pakete über UDP übertragen werden, werden sie bei Verlust nicht nachbestellt. Daher sind auch die Dropraten relevant, da bei zu hohen Dropraten die Gesprächsqualität merkbar schlechter wird. Hier sieht man nun, dass der UDP Verkehr einen sehr hohen RTP Paket Verlust auslöst. Der maximale zu akzeptierende Verlust beträgt 5% (siehe Kap. 2.1.2), welcher bei dieser Simulation, gut ersichtlich im Verkehrsgraphen (Abb. 21), bereits beträchtlich überschritten wurde.

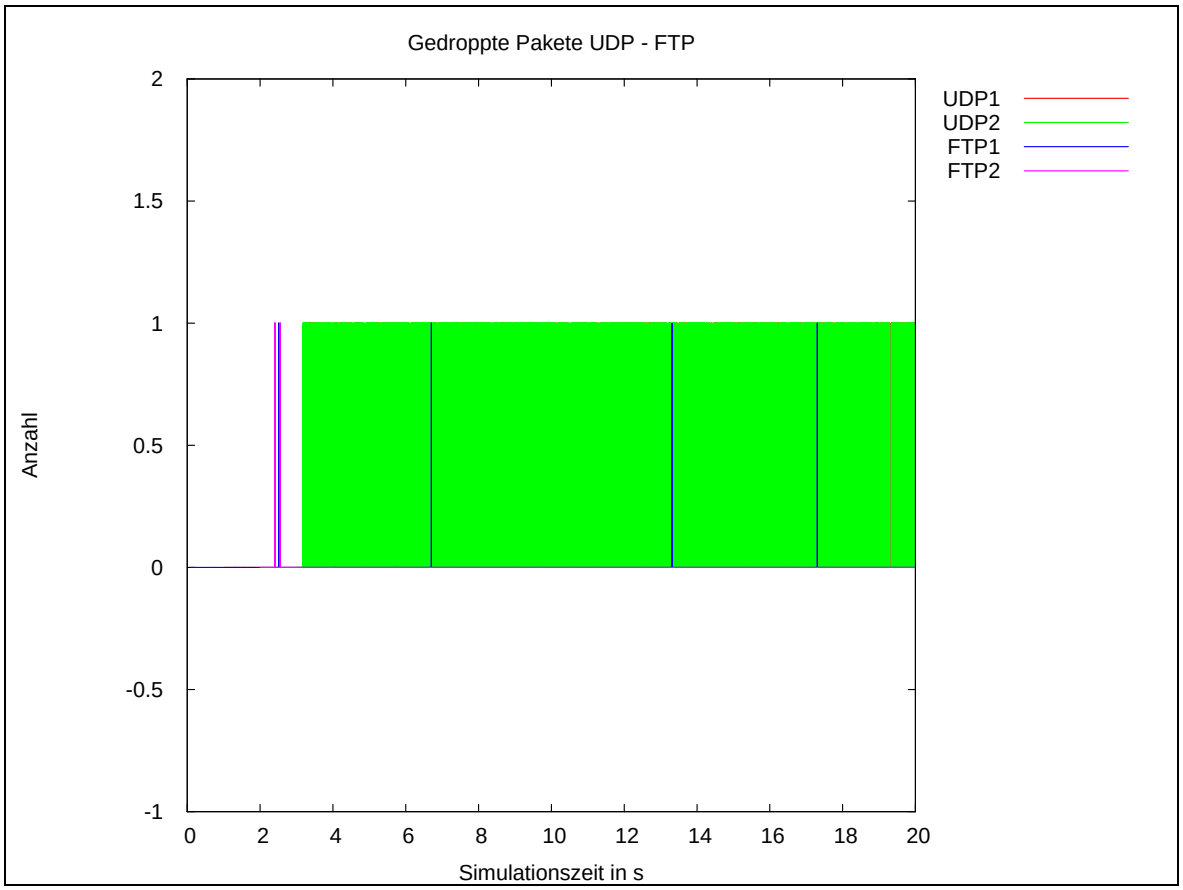

Abbildung 32 Gedroppte Pakete normal UDP – TCP

Man sieht anhand dieser Graphik, dass es neben dem RTP Verkehr hauptsächlich beim UDP Verkehr zu starkem Paketverlust kommt, da sie sich eine gemeinsame Warteschlange teilen. Der TCP Verkehr verliert nur bei Sekunde 3, 7, 13 und 18 Pakete (wie bereits beschrieben).

5.4. Simulation Integrated Services (RSVP)

In diesem Abschnitt wird die vorherige Teststellung an das RSVP angepasst und die Änderungen der Resultate werden erläutert. Der Simulator wird mittels eines Plugins um die Funktionen des RSVP erweitert und danach der Verkehr zwischen den beiden VoIP Telefonen mittels RSVP priorisiert.

Die neuen Parameter:

- Reservierte Bandbreite für RSVP: 0.1 (10%)
- RSVP Protokoll-Bandbreite: 50 bit (für Steuerdaten reserviert)

Es werden den Traffic-Flow-ID's des RTP Verkehrs eigene Sessions zugeordnet und die Pfadreservierung durchgeführt. Damit erkennt der NS-2 welche Übertragung mittels RSVP Unterstützung gesendet werden soll. Der NS-2 sendet automatisch beim Start der RTP Übertragung die Pfadreservierungen an die zwei Router, welche ihre Warteschlangen danach anpassen, um die RTP Pakete priorisiert durchzustellen.

5.4.1. Voice (RTP) Verkehr

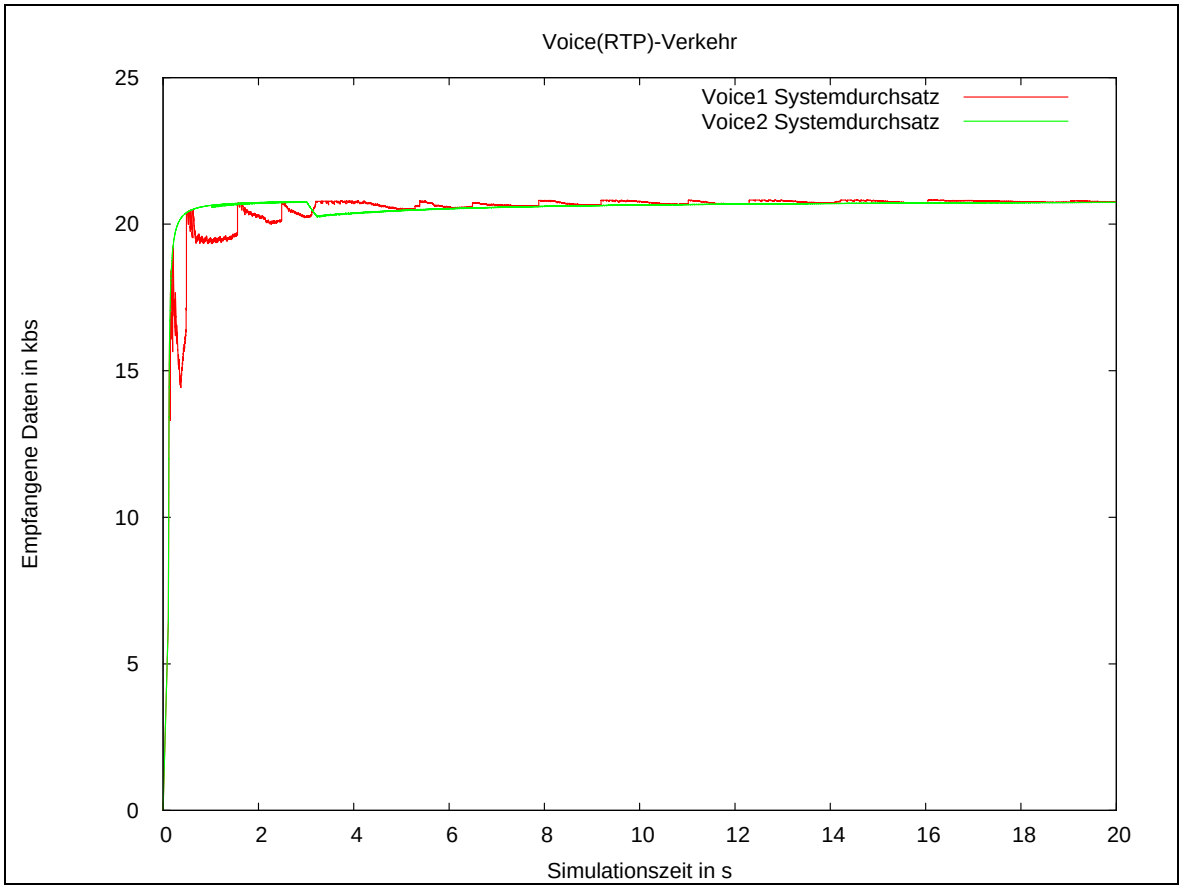

Abbildung 33 Voice (RTP) Verkehr RSVP

Im Gegensatz zur Simulation ohne RSVP sieht man, dass das nachdem alle Übertragungen gestartet wurden, eine sich asymptotisch an den Grenzwert von 20 kbit/s annähernde Bandbreitenkurve bei beiden VoIP Übertragungen ohne Bandbreitenabfall durch den UDP Verkehr entsteht. Man erkennt bei Sekunde 3, dem Moment wo der zweite UDP Verkehr zugeschaltet wird, einen kurzfristigen Einbruch, der durch das nun zum Tragen kommende RSVP Queue Management sofort wieder behoben wird. Es werden selektiv aus der Warteschlange die priorisierten Pakete der reservierten Ströme abgearbeitet, um die schon vorher reservierten Ressourcen nun zur Verfügung zu stellen. Ab diesem Moment pegelt sich auch der zweite VoIP Strom auf seine erforderliche Bandbreite ein.

Der Einbruch beim Start des ersten VoIP Verkehrs bleibt wie bei der vorherigen Simulation erhalten, da die TCP Pakete, die gleichzeitig versendet werden, zwar die Bandbreite nicht beeinflussen, aber trotzdem kurzfristig in der Warteschlange das Versenden der Voice Pakete verzögern.

5.4.2. TCP und UDP Verkehr

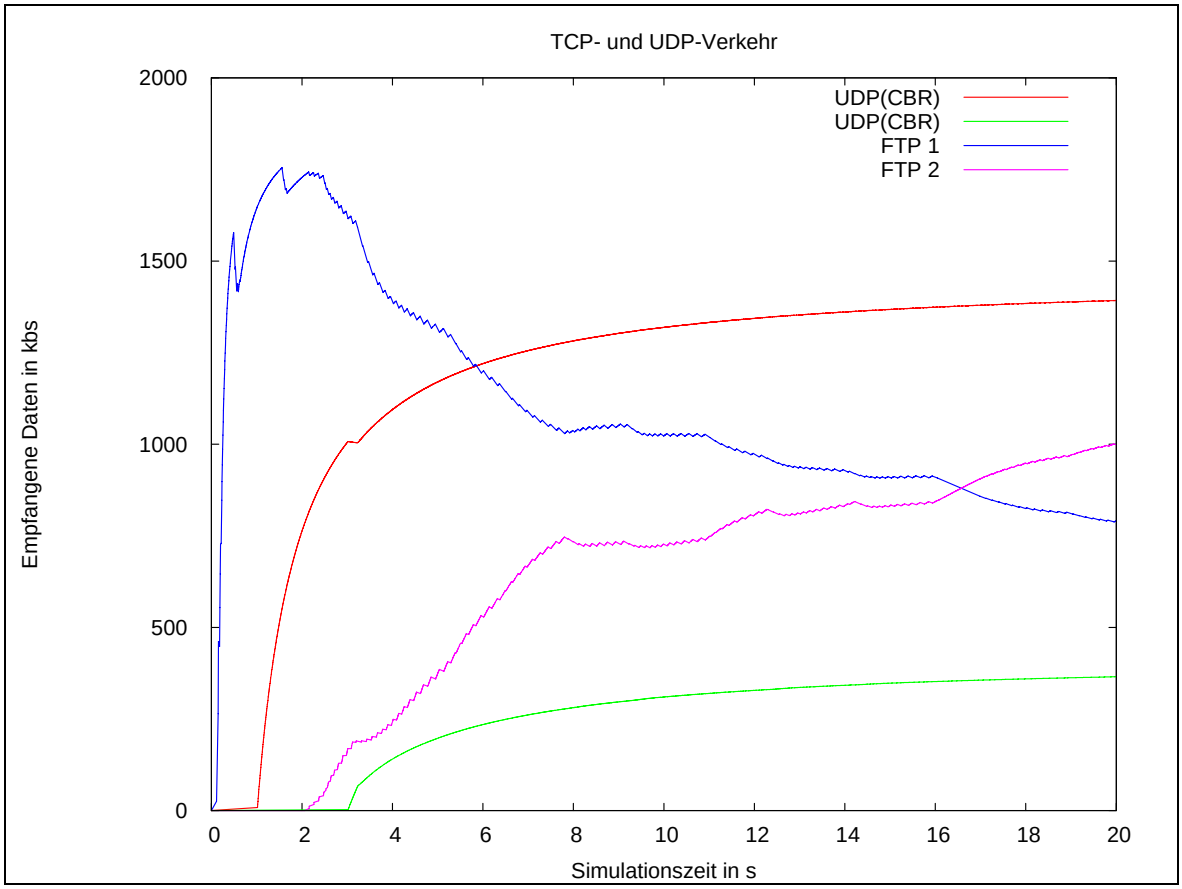

Abbildung 34 TCP und UDP Verkehr RSVP

Bei der Betrachtung der TCP- und UDP- Bandbreiten sieht man kaum eine Änderung. Allerdings wirkt sich die Bandbreitenreservierung des RSVP für den RTP Verkehr leicht glättend auf den restlichen Verkehr aus, da im Vorhinein schon weniger Gesamtkapazität verfügbar ist und

daher auch weniger Pakete durchgestellt werden können, woraus sich ein geringerer Anstieg der Bandbreitenkurven ergibt und somit eine schnellere Konvergenz eintritt.

5.4.3. Voice Delay

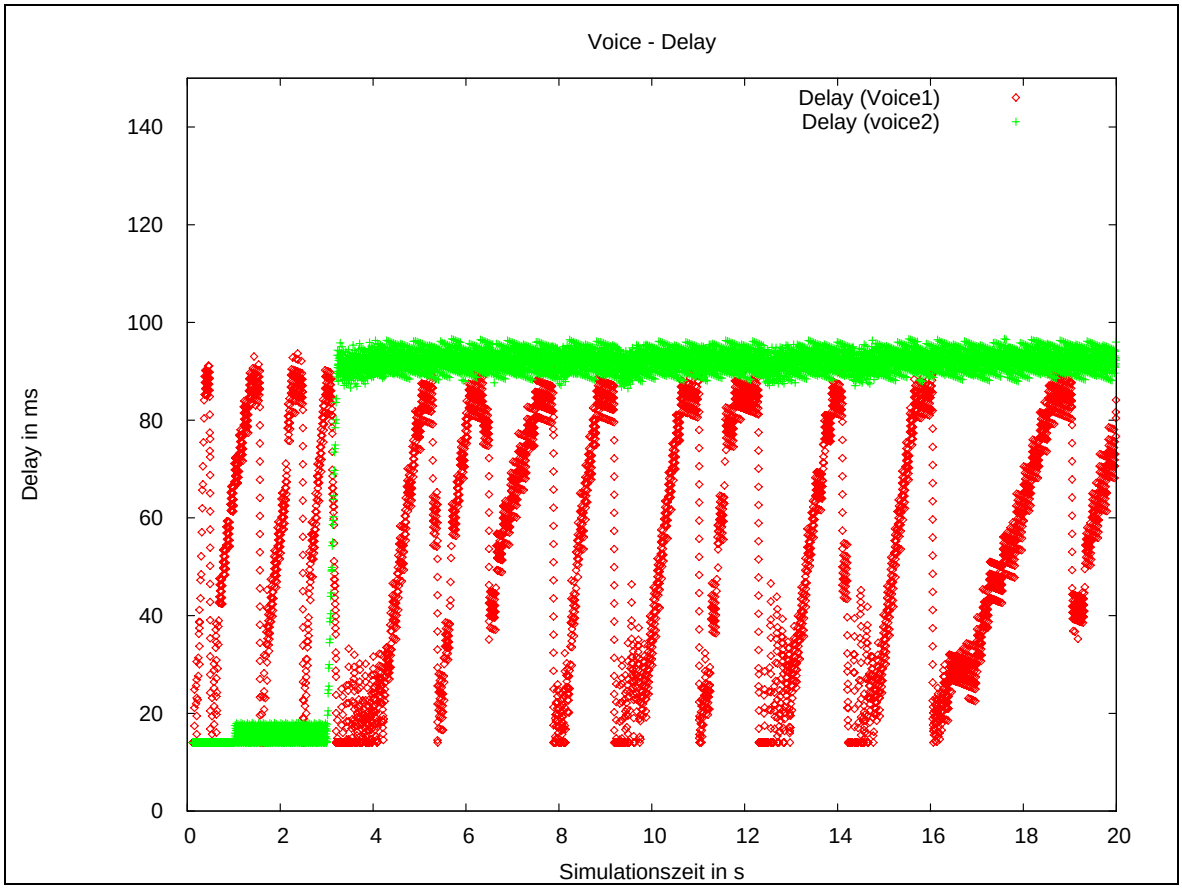

Abbildung 35 Voice Delay RSVP

Beim Voice Delay sieht man bereits eine deutliche Verbesserung. Der Maximalwert liegt, nachdem sich RSVP an alle auftretenden Datenströme angepasst hat, nun bei unter 100 ms; ohne RSVP lag er bei ca. 140 ms. Das ist das Resultat der Priorisierung des RTP Verkehrs in den Routerwarteschlangen. Somit ist ein geringerer Gesamtdelay möglich. Die Schwankungen des ersten VoIP Stromes entstehen durch

den durch RSVP modifizierten Queueingmechanismus, da dieser immer erst zu Wirken beginnt, wenn das Warteschlangenlimit erreicht wird. Im Gegensatz zum ersten VoIP Datenstrom geschieht das beim zweiten nicht, da durch die fehlenden Acknowledgements des UDP keine Drosselung der Paketsendefrequenz resultiert und die Warteschlange konstant gefüllt und auch das Delay konstant, mit weniger Streuung behaftet, aber immer noch verhältnismäßig hoch ist.

5.4.4. Voice Jitter

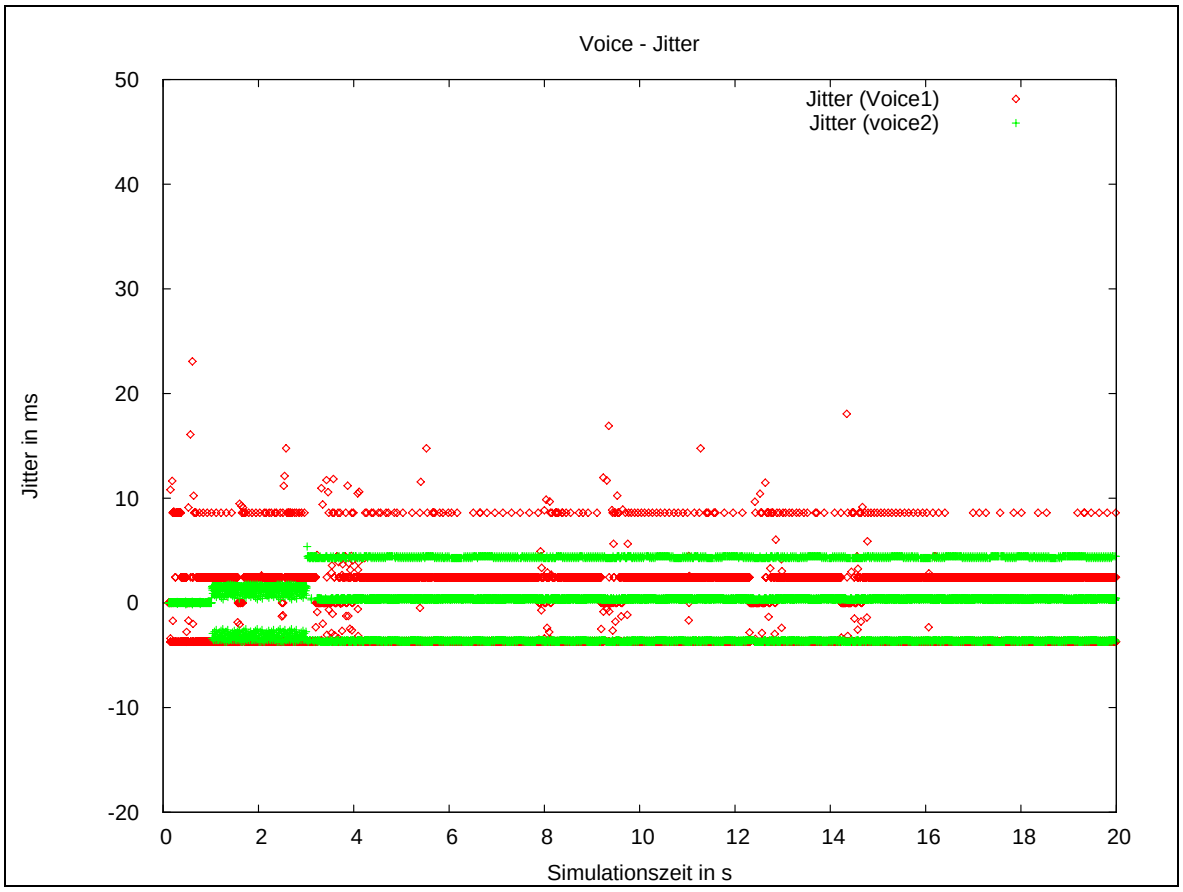

Abbildung 36 Voice Jitter RSVP

Auch beim Voice Jitter zeigt sich eine deutliche Verbesserung. Während ohne RSVP der Maximalwert bei 35 ms lag, fällt er nun unter 25 ms.

Der Großteil der Pakete schafft es unter 9 ms Laufzeitdifferenz, im Gegensatz zur den ehemals 21 ms. Somit ist es mit einem kleineren Jitterpuffer möglich, die Sprachqualität zu steigern.

5.4.5. Queuelängen

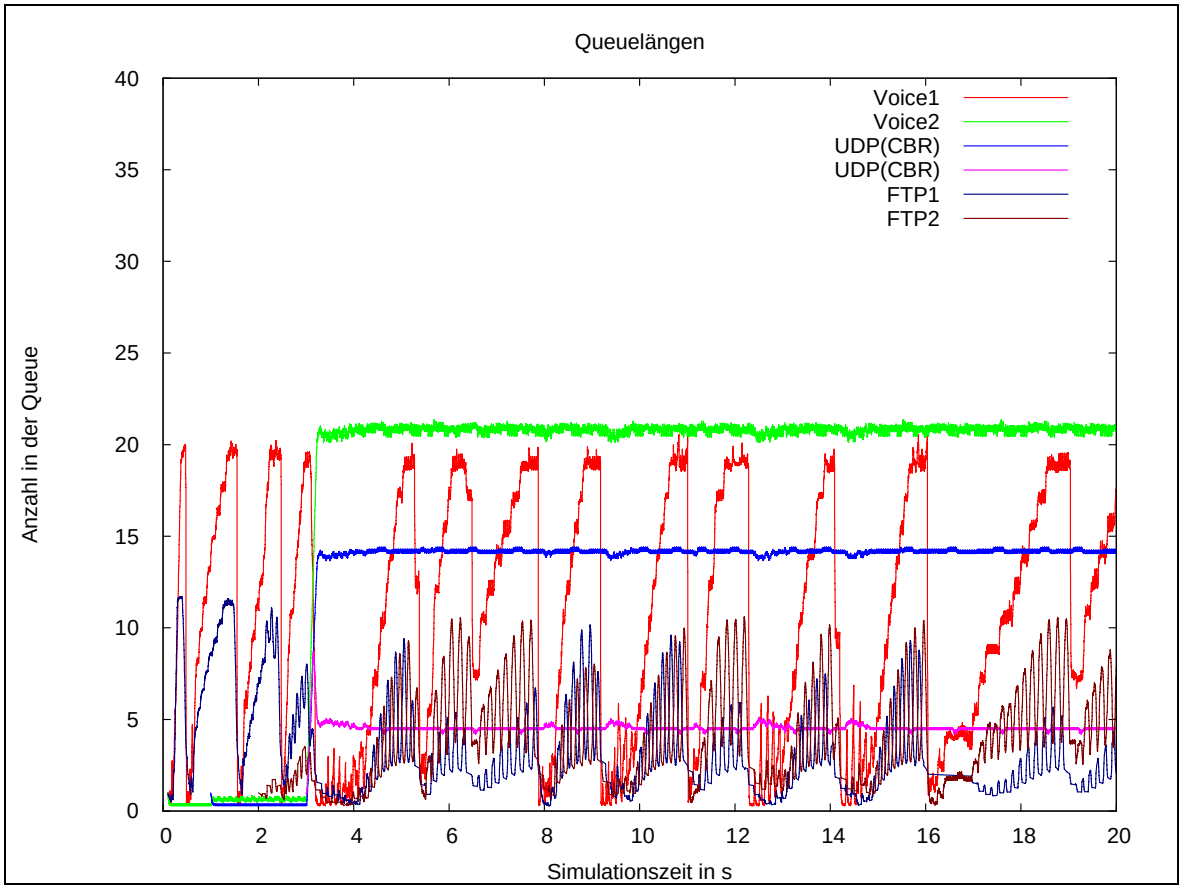

Abbildung 37 Queuelängen RSVP

Auf den ersten Blick ändert sich bis auf die Glättung der Kurven bei den mittels RSVP gesteuerten Routerwarteschlangen wenig, aber dennoch wird hier der RSVP Mechanismus sichtbar. Zu erkennen ist, dass die Warteschlange des zweiten Routers, welche die UDP Pakete abarbeitet, konstant an ihrem Limit arbeitet. Dies hat zur Folge, dass sich immer gleich viele Pakete von allen drei anfallenden Strömen in der

Warteschlange befinden, wobei die RTP Pakete garantiert verschickt werden (siehe „Gedroppte Pakete“; Abb.34). Die die Leistungsgrenze überschreitenden UDP Pakete werden zwar sofort verworfen, aber durch den kontinuierlich anhaltenden Paketfluss sofort wieder ersetzt, was eine konstante Anzahl an Paketen in der Warteschlange und eine hohe Verzögerung der RTP Pakete zur Folge hat.

5.4.6. Gedroppte Pakete

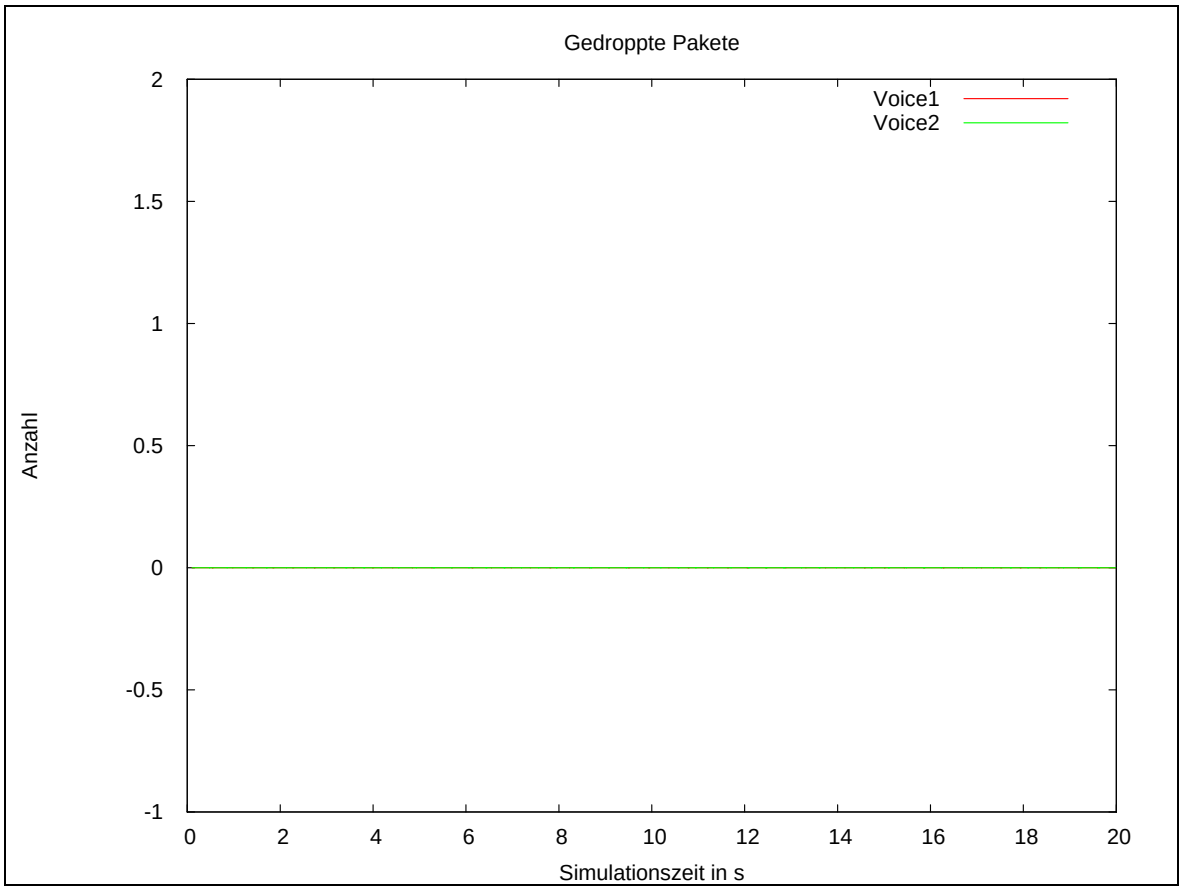

Abbildung 38 Gedroppte Pakete RSVP Voice

Alle RTP Pakete werden durch die Bandbreitenreservierung vollständig ausgeliefert und es zu keinem Paketverlust kommt.

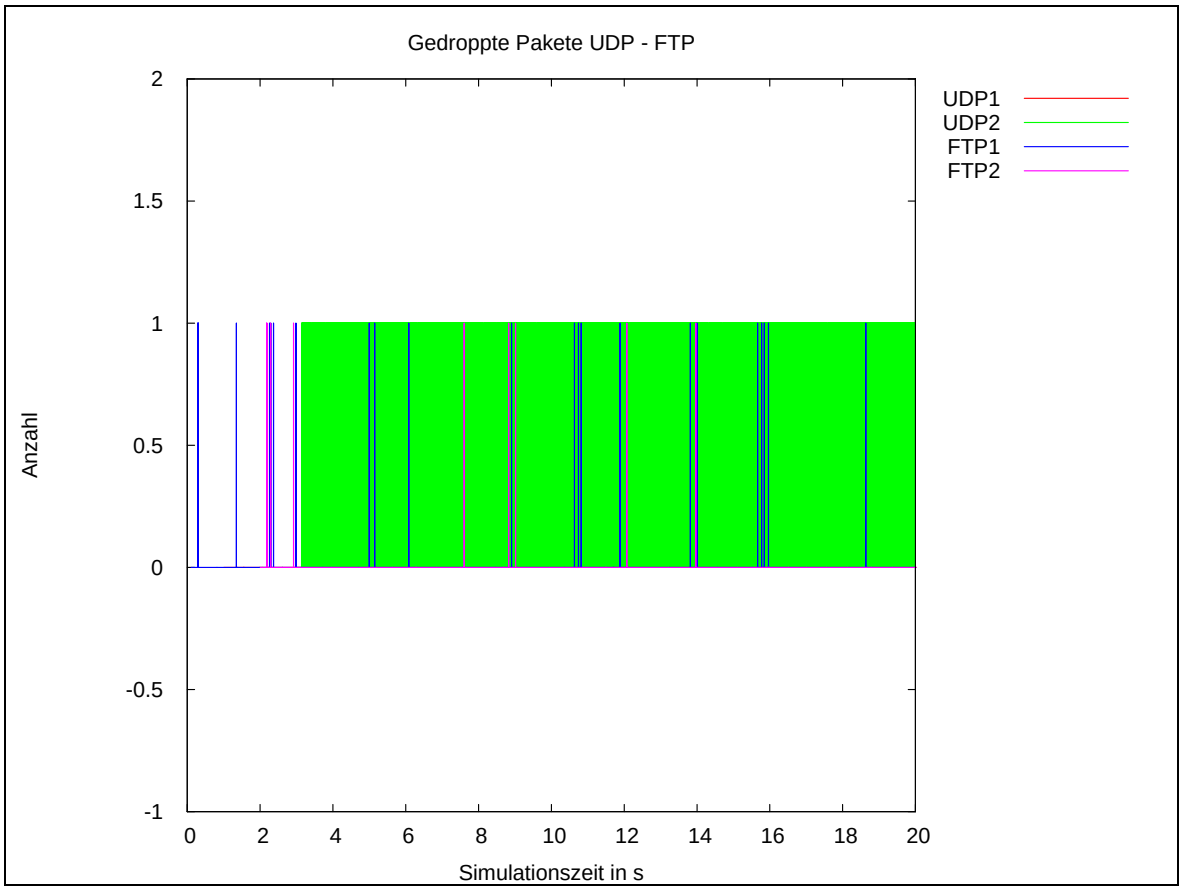

Abbildung 39 Gedroppte Pakete RSVP TCP - UDP

Ab dem Zeitpunkt, wo die Verbindung zwischen den Routern ihr Kapazitätslimit erreicht und RSVP aktiv wird, treten bei den nicht reservierten Verbindungen noch mehr Paketverluste ein, als dies schon vorher der Fall war, insbesondere bei den TCP Verbindungen.

5.5. Simulation Differentiated Services

Dieses Service ist bereits fix im NS-2 implementiert und trägt die Bezeichnung MRED (Multi Random Early Discard). Dieses Queuemanagement basiert auf der Markierung der verschiedenen Pakete, um den gewünschten Verkehr zu bevorzugen. Es werden mehrere virtuelle Queues für die verschiedenen QoS Klassen erzeugt. Je nach Markierung des Paketes gelangt es dann in eine eigene Queue. Diese Queues werden mit unterschiedlicher Priorität abgearbeitet. Wieviele Queues und welche Dienstgüten existieren, kann beliebig eingestellt werden.

Als Policier wird der srTCMPolicer (*Single Rate Three Color Marker*, RFC 2697) verwendet, welcher die Pakete mit 3 Farben markiert (rot, grün, gelb), um dem Router mitzuteilen, wie er die Pakete zu behandeln hat. Er markiert die Pakete aufgrund der CIR (*Committed Information Rate*), CBS (*Committed Burst Size*) und EBS (*Excess Burst Size*). CBS bedeutet eine kurzfristige Erhöhung des Datenflusses, um Spitzen abzufangen, ohne die CIR erhöhen zu müssen. Die EBS gibt zusätzlich die kurzfristige, maximal mögliche Datenrate an, falls ungenutzte Bandbreite vorhanden ist. Wenn die Pakete die CIR (definierte Bandbreite) nicht überschreiten, werden sie mit grün markiert. Zwischen gelb und rot unterscheidet dann die Einhaltung von CBS und EBS. Bei dieser Simulation kommt das weniger zur Anwendung, da die RTP Pakete aufgrund ihrer niederen Bandbreite grün markiert und dadurch bevorzugt werden. Der restliche Verkehr wird als gelb oder rot markiert, erhält also eine niedrigere Priorität. Da die Steuerung des MRED in NS-2 sehr komplex ist, wird an dieser Stelle nicht näher darauf eingegangen, sondern auf folgende Literatur verwiesen (siehe ALTMAN & JIMÉNEZ, 2003; siehe OPEN IP, 2000).

Die MRED Parameter:

- Reelle Queue: 1
- Virtuelle Queues: 3
- CIR: 2200 bit/s
- CBS: 200 bit
- EBS: 300 bit

5.5.1. Voice (RTP) Verkehr

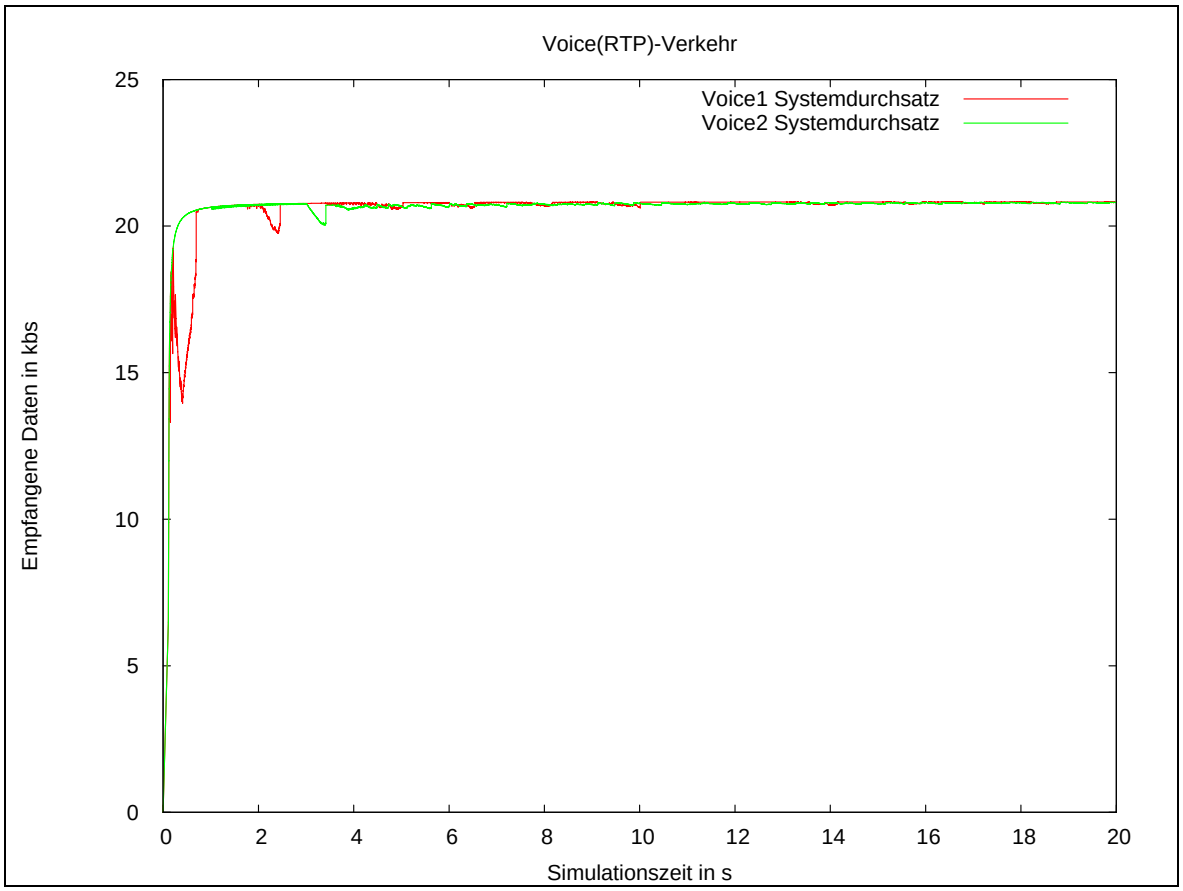

Abbildung 40 Voice (RTP) Verkehr Differentiated Services

Beim Voice (RTP) Verkehr sieht man bereits die Funktionsweise des MRED. Anfangs müssen die Router erst die virtuellen Queues zuordnen. Anders als bei RSVP, wo bereits vor der Übertragung dem Router per Steuerungsprotokoll mitgeteilt wird, wie er mit dem Verkehr umzugehen hat, muss hier der Router dies erst aus den ersten Paketen herauslesen und die neu hinzugekommenen virtuellen Queues befüllen und deren zugeordneten Pakete bewerten. Der Verkehr stabilisiert sich dennoch sehr rasch bei 20 kbit/s, bis auf die beiden kurzen Einbrüche bei Sekunde 2, respektive 3, wenn die zweiten TCP und UDP Ströme starten.

5.5.2. TCP und UDP Verkehr

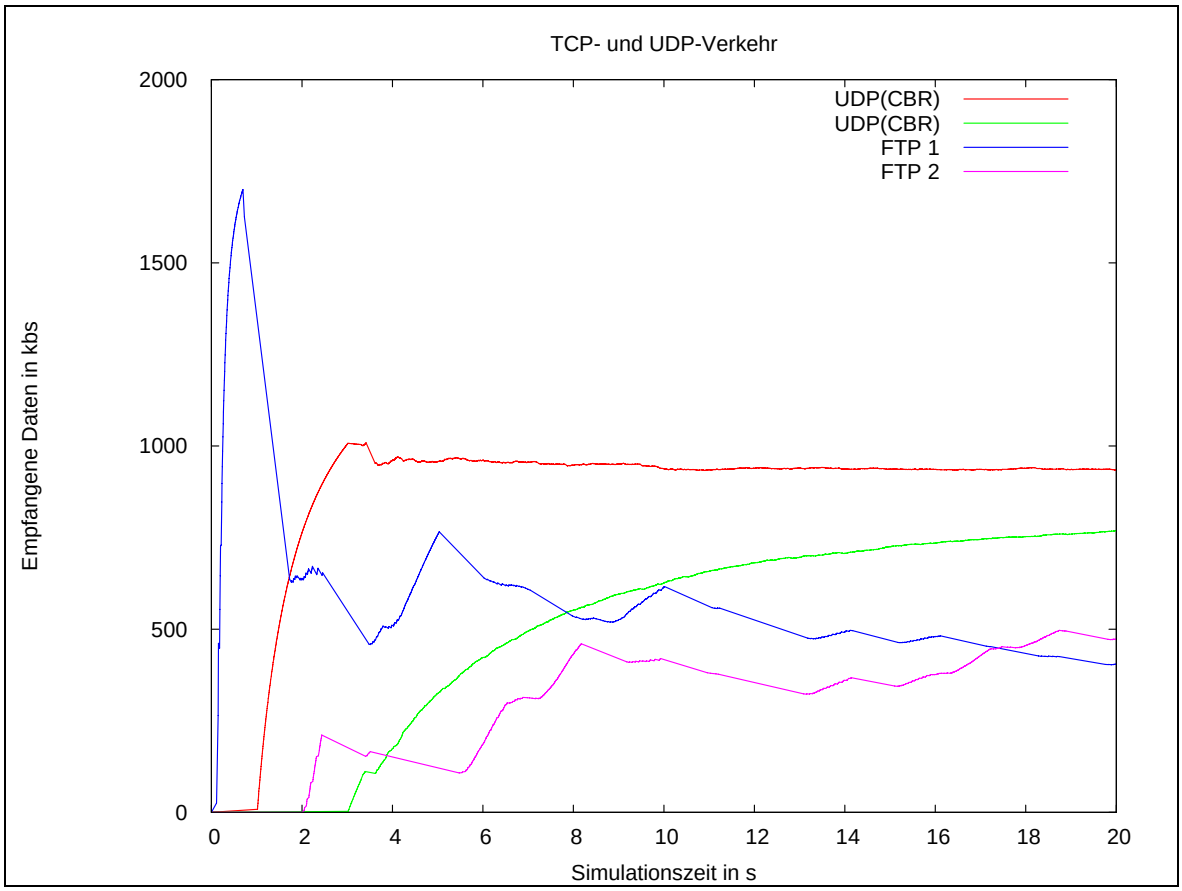

Abbildung 41 TCP und UDP Verkehr Differentiated Services

In dieser Grafik ist zu erkennen, dass bei den TCP Übertragungen starke Einbrüche auftreten, die durch den Markierungsalgorithmus verursacht werden. Da am Anfang der jeweiligen TCP Übertragungen sehr schnell die CIR, also das zulässige Maximum für eine bevorzugte Behandlung, überschritten wird, werden alle nachfolgend eintreffenden Pakete des jeweiligen Stromes sofort als minder priorisiert (gelb oder rot) markiert. Dies hat zur Folge, dass diese Pakete beim Erreichen des Kanallimits als erstes verworfen werden, und daher auch keine Acknowledgements der Gegenstelle zurückkommen. Die hohe Anzahl der verlorenen Pakete bewirkt eine Senkung der TCP Window Size von standardmässig 20 auf 1. Der Mechanismus greift so konsequent ein, dass er auch das letzte und somit für TCP wichtige Paket verwirft, wodurch der Retransmission Counter des TCP Stacks gestartet wird und bis dessen Ablauf keine TCP Pakete mehr gesendet werden. Danach erfolgt ein neuerlicher Sende-versuch, wobei sich das TCP auf die ausgelastete Kapazität der Verbindung zwischen den Routern neu einstellt und dementsprechend weniger stark sendet.

Bei UDP kommt es zu fast demselben Effekt, nur das UDP die Transmission nicht einstellt, sondern einfach durch den Markierungs-mechanismus beschränkt wird, wenn bei Sekunde 3 der Anstieg der Transferrate beim ersten UDP Verkehr abrupt abbricht und sich bei ca. 1 Mbit/s einpendelt.

5.5.3. Voice Delay

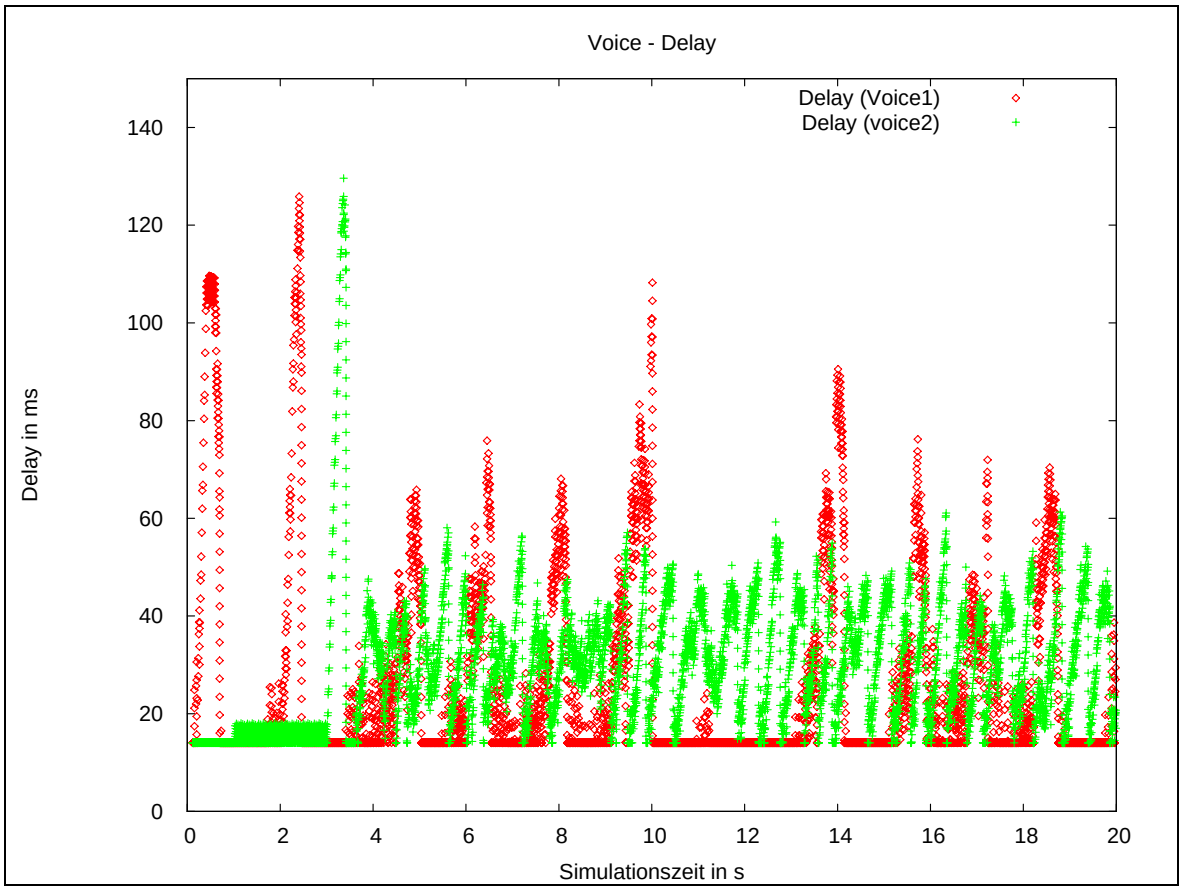

Abbildung 42 Voice Delay Differentiated Services

Die erste Verzögerungsspitze des Voice2 Verkehrs lässt sich auf die anfänglich unbewertete Warteschlange zurückführen, bei der der UDP Verkehr für einen kurzen Moment scheinbar die CIR Beschränkung einhält, diese aber gleich danach überschreitet und eine schlechtere Priorität erhält.

Derselbe Effekt hat bei dem Voice1 Verkehr hinsichtlich der TCP Datenströme eine ganz andere Ursache: Da der TCP Mechanismus sich durch unregelmäßig oder gar nicht ankommende Acknowledgements immer wieder selbst reguliert, fällt er bei dem Versuch, seine Transfer-rate zu erhöhen, zuerst unter die CIR Schranke, mit der Auswirkung,

dass er gleich gut wie der VoIP Verkehr behandelt wird, dann jedoch kann er diese Vorgaben nicht mehr erfüllen und wird herabgestuft. Das passiert in regelmäßigen Abständen abwechselnd einmal mit dem TCP 1 und dann mit dem TCP 2 Verkehr.

5.5.4. Voice Jitter

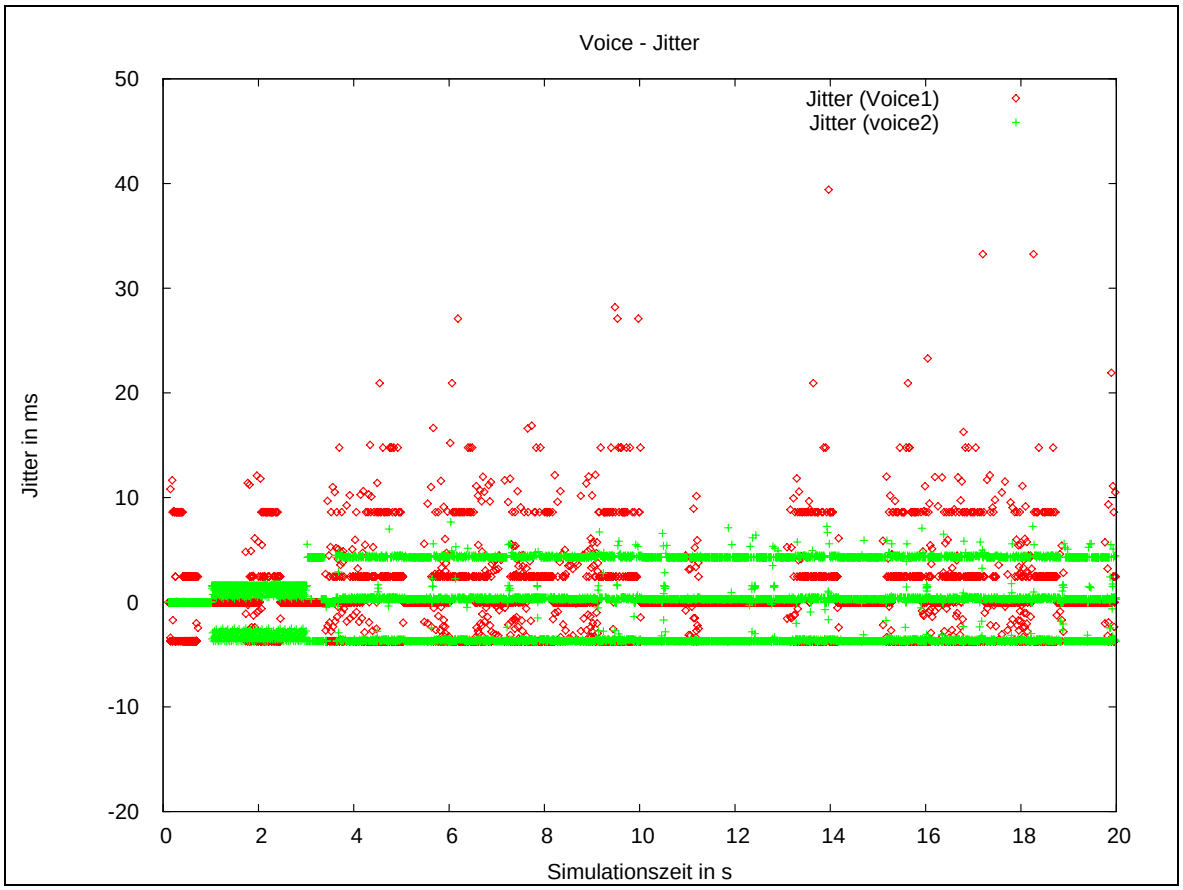

Abbildung 43 Voice Jitter Differentiated Services

Bei der Betrachtung des Jitters fällt die sehr geringe, durchschnittliche Verzögerung auf, die bei 9ms liegt. Jedoch ist die Streuung im Vergleich zu der RSVP Simulation stärker, wobei sie aber nicht die Ausmaße der Simulation ohne QoS Methoden annimmt.

5.5.5. Queuelängen

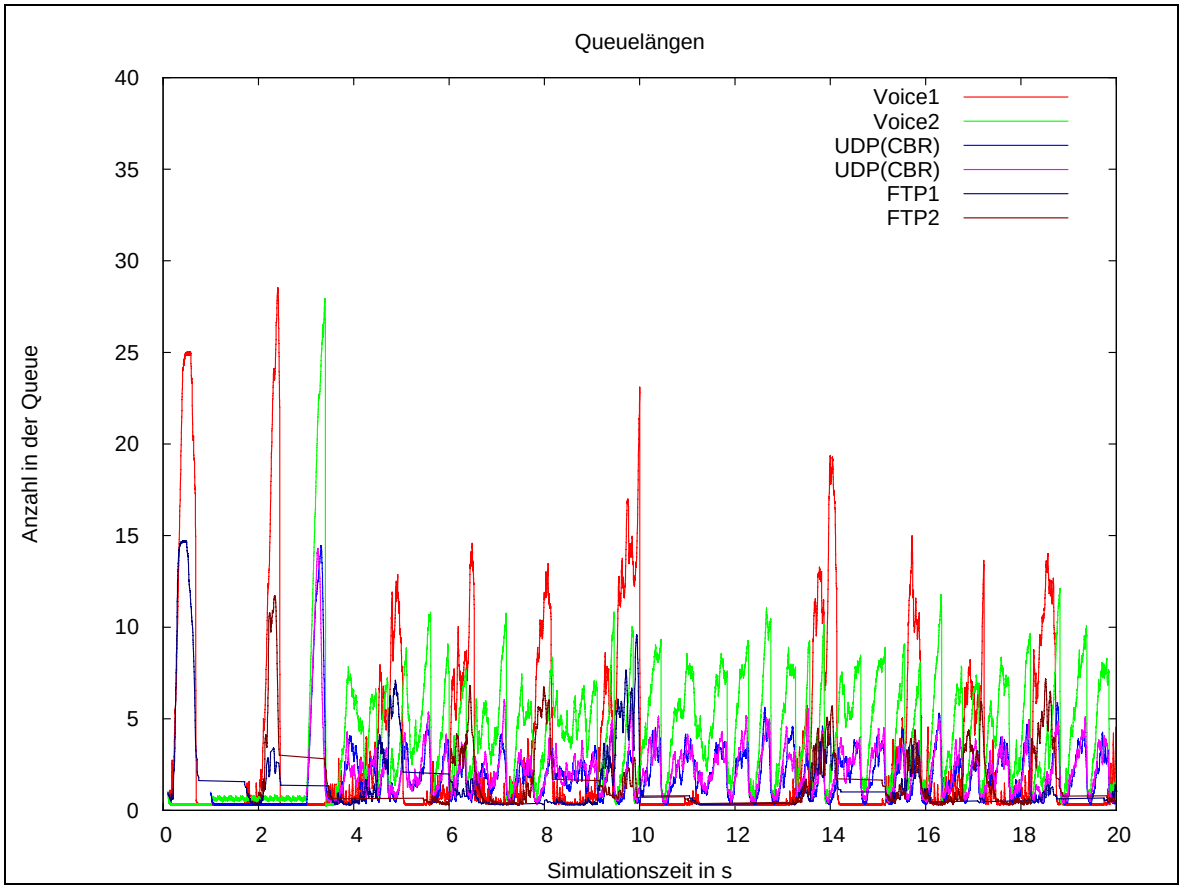

Abbildung 44 Queuelängen Differentiated Services

Anhand der dargestellten Queuelängen werden die zuvor beschriebenen Effekte nochmals gut sichtbar. Man kann erkennen, dass sowohl die Voice1 Pakete, als auch die TCP Pakete (FTP1 und FTP2) in der Warteschlange gereiht werden. Der RED Mechanismus verwirft alle neu zu reihenden Pakete ab dem Überschreiten des CIR Limits vor dem Eintreten in die Warteschlange und stellt deswegen bevorzugt die Voice1 Pakete durch.

5.5.6. Gedroppte Pakete

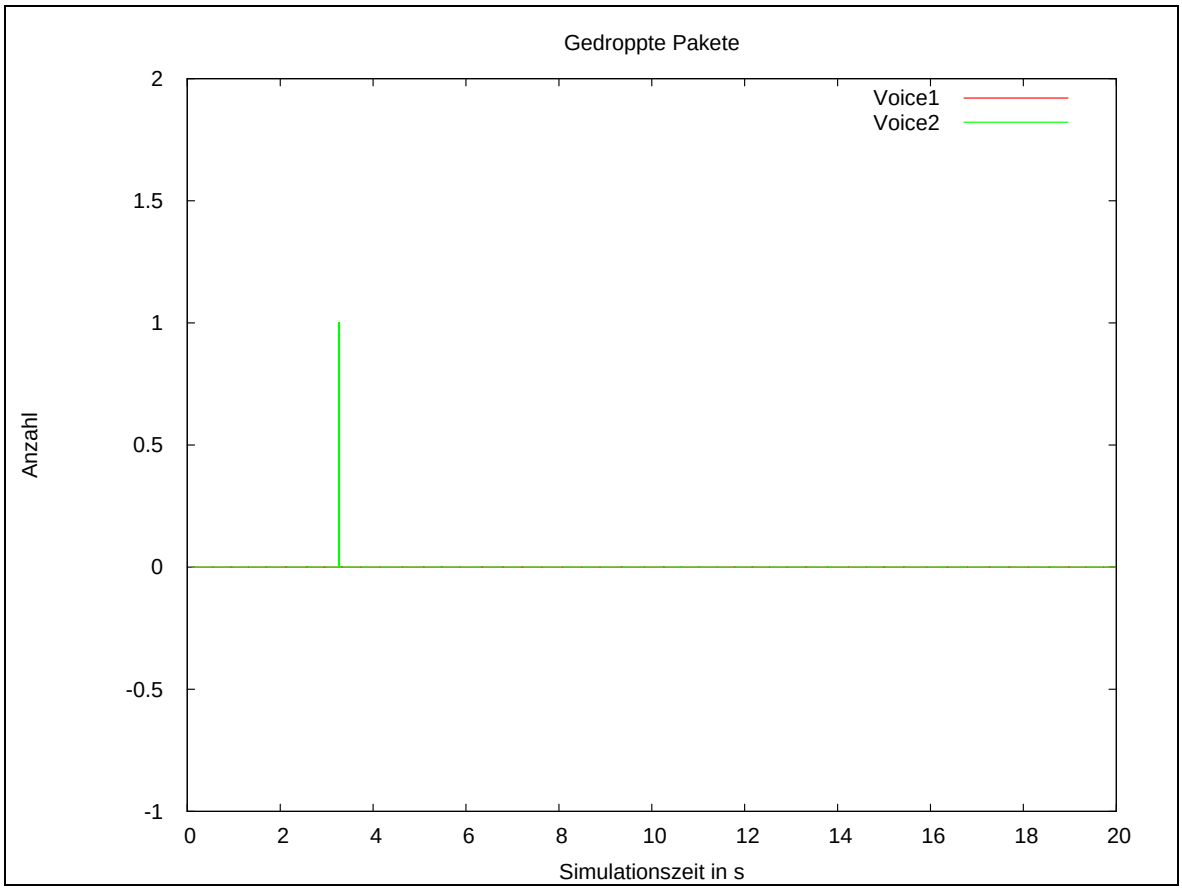

Abbildung 45 Gedroppte Pakete Differentiated Services Voice

Der einzige Paketverlust entsteht beim Einsetzen des zweiten UDP Streams, da kurzfristig der UDP 2 Verkehr die CIR Schranke einhält und der RED Mechanismus die Warteschlange leert, indem er zufällig Pakete verwirft, worunter sich auch RTP Pakete befinden. Eine genauere Analyse ergab einen Verlust von 3 RTP Paketen innerhalb von 20 ms, was unter dem Störungstoleranzwert von 5% liegt und somit kaum bis nicht mehr wahrnehmbar ist.

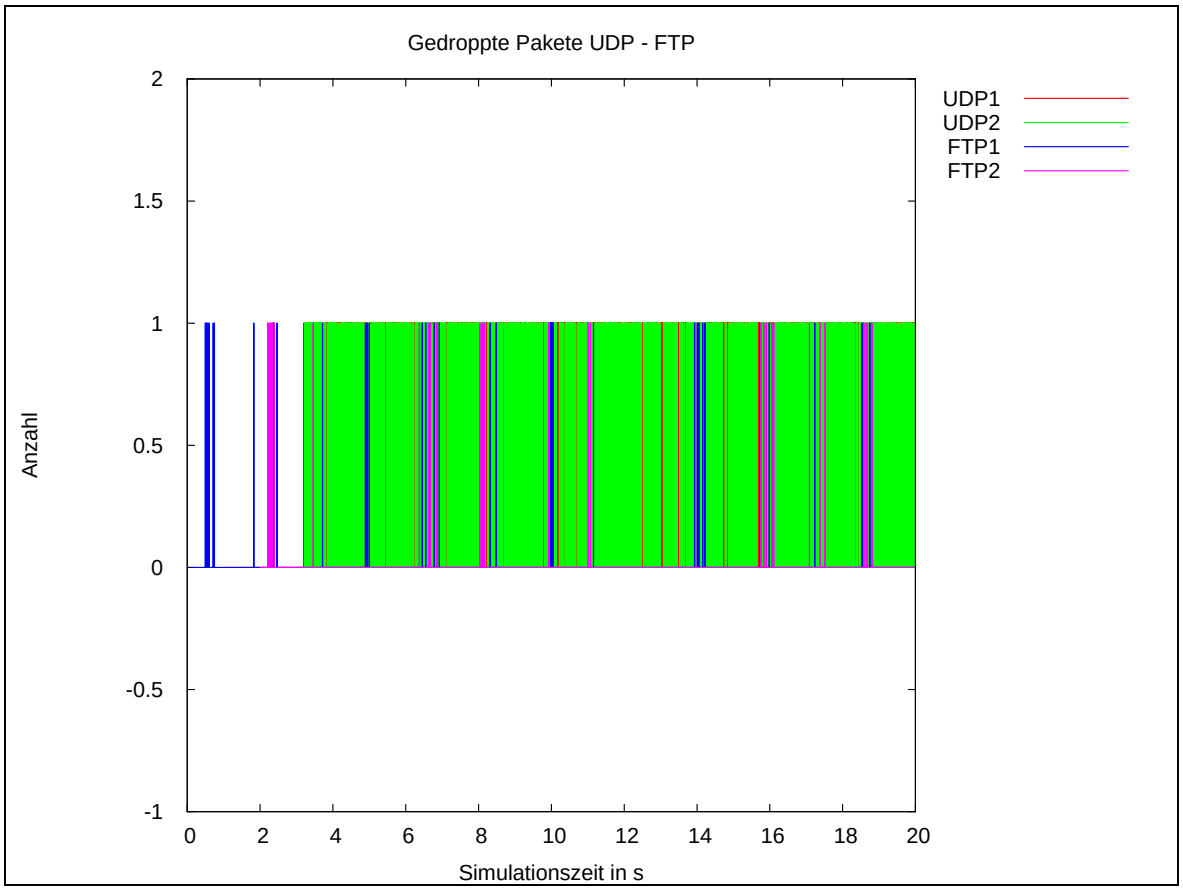

Abbildung 46 Gedroppte Pakete Differentiated Services TCP-UDP

Auch in Abb. 42 sieht man wieder die regelmäßig auftretenden Schwankungen des TCP Verkehrs wenn er nicht mehr die Beschränkungen der CIR Grenze erfüllt. Auch der starke UDP Paketverlust wird sichtbar, der in der Bandbreitengrafik (Abb. 37) zu einer Beschneidung des Gesamtdurchsatzes führt.

6. Fazit

6.1. Network Simulator NS-2

Mit dem NS-2 Simulator ist es möglich, nahezu alle erdenklichen Situationen im LAN zu simulieren. Aus Interesse wurden auch andere, nicht angeführte Szenarien durchgespielt, um die Möglichkeiten des Simulators auszutesten. Vor allem die Möglichkeit den Paketfluss grafisch darzustellen weckte Neugierde.

Der Berkeley NS-2 Simulator liefert realistische Aussagen über bestehende Computernetzwerke für VoIP Datenverkehr, weil die Ergebnisse der Simulationen durchaus mit den Ergebnissen im Labor vergleichbar sind. Darüber hinaus ermöglicht er die spezielle Untersuchung und Simulation mit implementierten QoS Mechanismen.

Da der Simulator mit Perl Scripten vieles automatisieren kann, sind auch große Netzwerke kein Problem.

6.2. QoS Methoden

Bei den QoS Methoden wird ersichtlich, dass sowohl die Integrated Services, als auch die Differentiated Services gut funktionieren. Man sieht deutlich, dass sich zeitkritischer Verkehr wie VoIP gut priorisieren lässt. Es ist anzunehmen, dass sich diese Technik problemlos auch auf große Netze übertragen lässt. Auch wenn mehr als zwei Router im Spiel sind, sollte QoS genauso gut funktionieren. Für kleinere Firmennetze wird wohl RSVP besser geeignet sein, da es weniger Anforderungen an die Hardware stellt, allerdings ist auch mit mehr *Overhead* zu rechnen.

Für große Unternehmen, welche mehrere Standorte via Internet verbunden haben, sind die Differentiated Services sicher die beste Wahl. Vor allem, da sich die SLAs mit den Providern koppeln lassen. Interessant ist sicher auch die Möglichkeit, diversen Anwendungen verschiedene Serviceklassen zuzuordnen. Allerdings erfordert dieses System einiges an Konfigurationen an den Routern.

6.3. Nächste Schritte

Interessant wäre es, auf die verschiedenen Arten der Differentiated Services näher einzugehen, da dieses Thema Stoff für weitere Nachforschungen liefert. Auch die Möglichkeit des NS-2, Funknetzwerke zu simulieren, stellt im Bezug auf QoS interessante Möglichkeiten dar.

Abkürzungsverzeichnis

- A/D - Analog / Digital
- ARPA - Advanced Reserch Projects Agency
- ATM - Asynchronous Transfer Mode
- CBR - constant bit rate
- CBS - Committed Burst Size
- CC – Contributing Source Identifikation Count
- CIR - Committed Information Rate
- CS-ACELP - conjugate structure algebraic code excited linear prediction
- CSMA/CA - Carrier Sense Multiple Access with Collision Avoidance
- CSMA/CD - Carrier Sense Multiple Access with Collision Detection
- D-ITG - Distributed Internet Traffic Generator
- DNS – Domain Name System
- DSCP - Differentiated Service Code Point
- DSL – Digital Subscriber Line
- EBS - Excess Burst Size
- EWMA - Exponential Weighted Moving Average
- FIFO - First In First Out
- HTTP - Hyper Text Tansfer Protocol
- IETF - Internet Engineering Task Force
- IP – Internet Protocol
- IPv6 - Internet Protocol Version 6
- ISDN – Integrated Services Digital Network
- ITU - International Telecommunication Union
- ITU-T - Telecommunication Standardization Bureau
- LAN - Local Area Network
- LPC - Linear Predictive Codes

- MRED - Multi Random Early Discard
- NAT - Network Adress Translation
- OSI - Open System Interconnection
- OSPF - Open Shortest Path First
- PCM - Pulse Code Modulation
- PHB - Per Hop Behavior
- QoS – Quality of Service
- RAS – Remote Access Service
- RED - Random Early Detection
- RFC - Request for Comment
- RIP - Routing Information Protocol
- RSVP - Resource Reservation Protocol
- RTCP - Realtime Transport Control Protocol
- RTP – Realtime Transport Protocol
- SIP – Session Initiation Protocol
- SLA - Service Level Agreement
- srTCMPolicer - Single Rate Three Color Marker
- SSRC – Sychronization Source Identifier
- SSRO – Synchronization Source
- TCA - Traffic Conditioning Agreement
- TCP - Transmission Control Protocol
- TCP/IP - Transmission Control Protocol / Internet Protocol
- TOS - Type of Service
- UC Berkeley - University of California at Berkeley
- UDP - User Datagram Protocol
- URL - Uniform Resource Locator
- VoIP – Voice over Internet Protocol
- WAN – Wide Area Network
- WLAN - Wireless Local Area Network

Abbildungsverzeichnis

Literaturverzeichnis

- **ALTMAN**, E. **& JIMÉNEZ**, T.: NS Simulator for beginners, Lecture notes, 2003, Universität de Los Andes, Mérida, Venezuela and ESSI, Sophia-Antipolis, France
- **COMER**, D.E.: Internetworking with TCP/IP, 1995, Prentice Hall, ISBN 0-13-018380-6.
- **FALL**, K. **& VARADHAN**, K.: The *ns* Manual, The VINT Project, A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2005
- **HOGREFE**, D.: Mobilkommunikation 2, Voice over Wireless LAN, Universität Göttingen, Skriptum 2005
- **HUNT**, C.: TCP/IP Network Administration, 1992, O'Reilly & Associates Inc, ISBN 1-56592-322-7.
- **HUTTER**, S.: Verknüpfung von Quality of Sevice (QoS) und Routingmethoden in IP-Netzen, 2000, Technische Universität Wien
- **IBM TCP/IP**, Murhammer M., Atakan O., Bretz S., Pugh. L., Suzuki K, Wood D.: TCP/IP Tutorial and Technical Overview,1998
- **ITU**: http://www.itu.int/home/index.html, Stand Juli 2005
- **JOHNSTON**, A.B.: SIP - Understanding the Session Initiation Protocol, 2004, Artech House, ISBN 1-58053-655-7
- **KÖHLER**, R.D.: Voice over IP, 2002, ISBN 3-8266-4067-5
- **LINUXFIBEL**: „Die Geschichte des Internet", www.linuxfibel.de
- **LUTZ**, A.: Technologische, organisatorische und wirtschaftliche Konzeption für die Sprach-Daten-Integration auf der Basis von Voice over IP an der Technischen Universität Ilmenau, 2002, Technische Universität Ilmenau; Projektarbeit
- **MEINCKE**, M.: QoS-Aspekte bei der Einführung von VoIP, 2001, Eduard Siemens Institut für Allgemeine Nachrichtentechnik,
- **MOSIG**, A.: Net Simulator 2, Routing Seminar, 2004
- **OPEN IP**, Pieda P., Ethridge J., Baines M., Shallwani F.: A Network Simulator, Differentiated Services Implementation, Open IP, Nortel Networks, 2000
- **PETERSON**, L. **& DAVIE**, B.: Computer Networks, A Systems Approach, 2000, Morgan Kaufmann, ISBN 1-55860-514-2.

- **RFC 791**: Internet Protocol, 1981, DARPA Internet Programm
- **SCHULZRINNE**, H. **& CASNER**, S. **& FREDERICK**, R. **& JACOBSON**, J.: RTP: A Transport Protocol for Real-Time Applications, IETF, 2001.
- **STALLINGS**, W.: Data and Computer Communications, 1997, Prentice Hall, ISBN 0-13-571274-2.
- **STILLER**, B.: Quality-of–Service, Dienstgüte in Hochleistungsnetzen, 1995, ISBN 3-8266-0171-8
- **TANENBAUM**, A.S.: Computer Networks, 2003, ISBN 0-13-038488-7
- **TRUMMER**, B.: Inter-Domain Clearing in der IP-Telefonie unter Berücksichtigung von Sicherheitsaspekten, 2001, Technische Universität Graz; Diplomarbeit
- **VOCALTEC**: http://www.vocaltec.com, Stand Mai 2005
- **WALKE**, B: Mobilfunknetze und ihre Protokolle Band 1&2, 2001, ISBN 3-519-26430-7
- **WIKIPEDIA**: http://de.wikipedia.org, Suchbegriff: IPv4