Anwendungsbezogene Dienstgüte-Emulation drahtloser Kommunikationskanäle auf Ethernetbasis

Dipl-Inf. (FH) Michael Brüning, Prof. Dr.-Ing. Clemens Westerkamp, Fachhochschule Osnabrück
Email: {bruening, c.westerkamp}@fhos.de

Kurzfassung

Verteilte und mobile Anwendungen erfahren in den letzten Jahren zunehmende Verbreitung in allen Bereichen des alltäglichen Lebens. Dies schafft den Bedarf für neue Technologien zur drahtlosen und drahtgebundenen Kommunikation, deren Erprobung zur Gewährleistung einer hinreichenden Anwendungsqualität unerlässlich ist. Oftmals ist ein Test mit neuen Geräten und Systemen aufgrund geringer Verfügbarkeit und hoher Kosten jedoch unmöglich. Hier kann die Emulation der benötigten Systeme Abhilfe schaffen. Dieser Beitrag befasst sich mit der Beschreibung eines an der Fachhochschule Osnabrück entwickelten Dienstgüteemulators. Der erst Abschnitt führt kurz in das Thema ein und erläutert den Begriff Dienstgüte. Im zweiten Teil werden Konzept und Implementation der Emulation näher erläutert. Abschnitt 3 beschreibt die in der Anwendung gesammelten Erkenntnisse, bevor Abschnitt 4 die Ergebnisse zusammenfasst und Ausblick auf geplante Entwicklungen gibt.

1 Einleitung

Verteilte und mobile Anwendungen erfreuen sich in den letzten Jahren zunehmender Beliebtheit in nahezu allen Bereichen des täglichen Lebens, von Anwendungen in der Produktion, Logistik und Verwaltung bis hin zur privaten Kommunikation und Unterhaltung Die immer stärkere Vernetzung der eingesetzten Systeme erhöht die Anforderungen an die unterliegenden Netzwerke und begünstigt und bedingt die Entwicklung neuer Technologien zur drahtgebundenen oder drahtlosen Kommunikation. Diese werden zunächst in Standards definiert und sukzessive umgesetzt.
Die erreichbare und die reale Dienstgüte dieser neuen Technologien sind dabei von etlichen Einflussfaktoren abhängig. Für Dienste- und Anwendungsentwickler ist außerdem offen, welche Einschränkungen ihre Lösungen bei suboptimaler Dienstgüte erfahren. Zur Gewährleistung einer für die geplanten Anwendungen hinreichenden Qualität sind folglich Tests unter realen Anwendungsbedingungen nötig. Häufig stehen jedoch Simulationen und Netze oder Testcenter noch nicht in ausreichendem Maße und mit der wünschenswerten Variationsbreite zur Verfügung oder sind zu aufwändig in der Anschaffung. Hier kann die Emulation einzelner Netzelemente oder eines ganzen Netzwerkes einschließlich der zugehörigen Dienstgüteeigenschaften Abhilfe schaffen.
Der vorliegende Beitrag befasst sich mit der Beschreibung eines an der Fachhochschule Osnabrück entwickelten Emulators der Dienstgüte von IP-basierten drahtlosen Netzwerken.

1.1 Dienstgüte

Mit dem Begriff Dienstgüte sind im oftmals mehrere, teils nicht klar umrissene Assoziationen verbunden. Zum einen wird mit der Dienstgüte die vom Anwender erlebte Güte einer Anwendung bezeichnet, die sich nur unscharf definieren lässt und in der Regel durch die statistische Auswertung von repräsentativen Tests mittels einer vorher definierten Kennzahl gemessen wird. Zum anderen gibt es Definitionen, die Dienstgüte als den Grad der Erfüllung messbarer Qualitätsanforderungen beschreiben (vgl. **[1], [2]**).
In der vorliegenden Anwendung wird die Dienstgüte als eine Menge von Merkmalen betrachtet, welche die Qualität der Zusicherung der Netzwerkverbindung beschreiben, die einem Benutzer angeboten wird. Dieser Benutzer muss nicht zwangsläufig durch eine Person dargestellt werden, sondern kann ebenso gut ein Agent oder ein Service auf einer höheren Protokollebene sein (vgl. [3]). Allgemein kann die Dienstgüte daher auf verschiedenen Ebenen betrachtet werden, wie Abbildung 1 veranschaulicht.

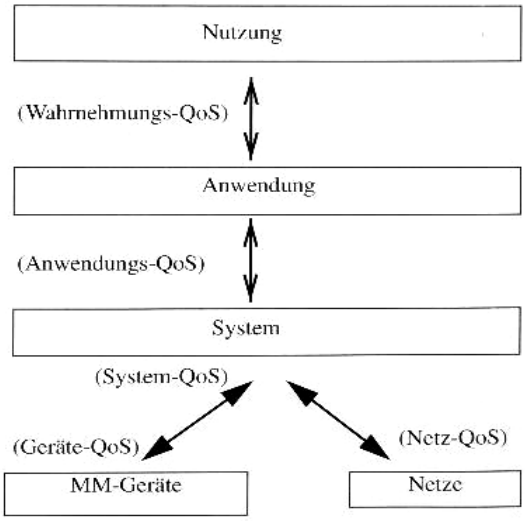

Abbildung 1 Verschiedene Arten von Dienstgüte[1]

Die Arbeit von Freitag [4] analysierte die für die Dienstgüte von IP-basierenden Netzen relevanten Parameter, von denen die folgenden in die Implementation des vorliegenden Emulators integriert wurden:

- Durchsatz (Throughput)
- Paketverzögerung (Delay)
- Varianz der Paketverzögerungen (Jitter)
- Paketverlust (Packet-Loss)
- Zusammenhängender Paketverlust (Packet-Loss-Bursts)

Diese Parameter sind indirekt aus anderen Definitionen wie zum Beispiel durch Paxson et al. [5] definierten *IP Performance Metrics* abgeleitet.
Westerkamp et al. [6] führten Messungen und Analysen dieser Parameter in realen Anwendungsszenarien mit drahtgebundenen und drahtlosen Netzwerken wie z.B. Wireless LAN und ISDN durch. Aus diesen Arbeiten ging eine Reihe von Dienstgüteprofilen für die in den Messungen eingesetzten Systeme hervor, die als Grundlage für Emulationen genutzt werden.

2 Konzept und Umsetzung

2.1 Konzept zur Emulation

Das dem Emulator zugrunde liegende Konzept besteht darin, eine Ethernetverbindung zwischen zwei Endpunkten eines Netzes über einen dritten Rechner umzuleiten, der mit zwei Netzwerkadaptern ausgestattet ist. Dieser nimmt die Emulation der im ersten Abschnitt angeführten Parameter vor, das heisst, die gesamte IP-Kommunikation zwischen den beiden Endpunkten wird gefiltert und die Pakete werden auf verschiedene Warteschlangen gesetzt, um Bandbreitenbeschränkungen, Verzögerungen und Jitter zu emulieren. Die dabei verwendeten Parameter werden in Form von netzspezifischen Profilen für anwendungsspezifische Paketgruppen (z.B. Steuerungs- und Netzpakete in H.323) durch Analyse erfasst. Des Weiteren können IP-Pakete verworfen werden. Abbildung 2 verdeutlicht dieses Prinzip.

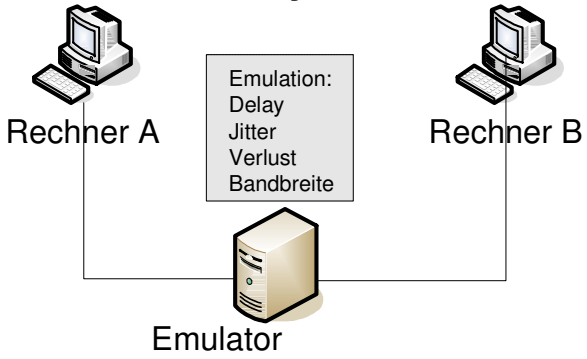

Abbildung 2 Emulationskonzept

Im Folgenden werden die Abläufe zur Realisierung der einzelnen Eigenschaften kurz näher erläutert (siehe auch Abbildung 3).

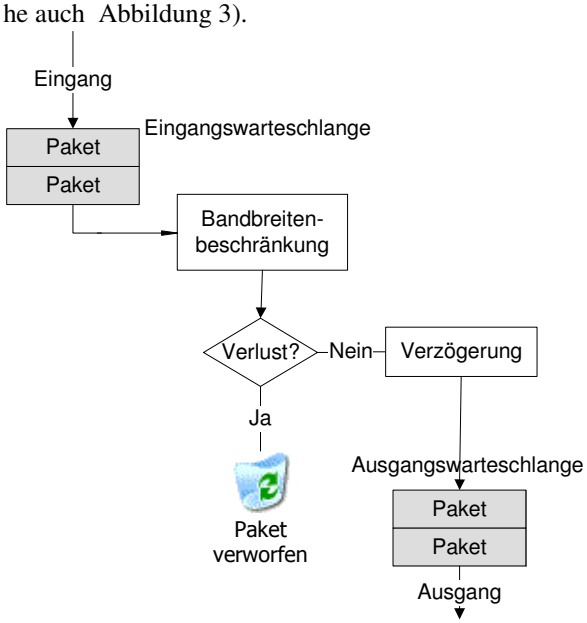

Abbildung 3 Ablauf des Emulationsalgorithmus

1. Zunächst werden die Pakete mit einem Zeitstempel versehen und ein Verzögerungswert aus der Standardverzögerung +/- eines innerhalb des eingestellten Jitter liegenden, zufällig ermittelten Wertes gesetzt. Dann wird das Paket auf eine Eingangswarteschlange gesetzt.
2. Es wird geprüft, ob die Bandbreite bereits voll ausgeschöpft wurde:
   a) Falls ja, verbleibt das Paket auf der Warteschlange, bis es gesendet werden kann.
   b)Falls nein, geht es weiter mit 3.
3. Es wird mittels einer Wahrscheinlichkeitsfunktion überprüft, ob das Paket verworfen werden muss:
   a) Ist dies der Fall, wird das Paket verworfen.
   b) Ist dies nicht der Fall, wird das Paket auf die Ausgangswarteschlange gesetzt.
4. Hier bleibt es, bis der gesetzte Verzögerungswert erreicht wurde und wird dann weiter gesendet.

Zu beachten ist, dass diese Vorgänge getrennt für jede Netzwerkkarte ablaufen, um Hin- und Rückkanal korrekt zu trennen und somit die Vollduplexfähigkeit zu erreichen. Wichtig ist auch, dass die Funktion zum Verwerfen von Paketen auch burst-artige Fehler unterstützt.

2.2 Umsetzung des Konzeptes

Als Basis für die Implementation des Emulators wurde nach einem Vergleich mehrerer Alternativen das freie Betriebssystem Linux verwendet. Diese Wahl begründet sich in der Hauptsache dadurch, dass Linux eine Vielzahl von Netzwerkprotokollen im Betriebssystemkern, dem so genannten *Kernel*, unterstützt und im Rahmen der GPL (siehe [7]) frei verfügbar ist und im Quelltext vorliegt. Daher lässt sich der Kernel mit fest integrierten oder zur Laufzeit dynamisch ladbaren *Kernel-Modulen* erweitern.
Des Weiteren bietet Linux sehr gute Möglichkeiten zum Filtern des auf den Netzwerkschnittstellen auftretenden Verkehrs durch das *Netfilter* API, das fester Bestandteil des Kernels ist und auf dem auch die in Linux integrierte Firewall iptables basiert. In Netfilter können so genannte *Hooks* registriert werden, um IP-Pakete an verschiedenen Stellen im Routingprozess zu filtern. Abbildung 4 zeigt eine vereinfachte schematische Übersicht dieses Ablaufs.

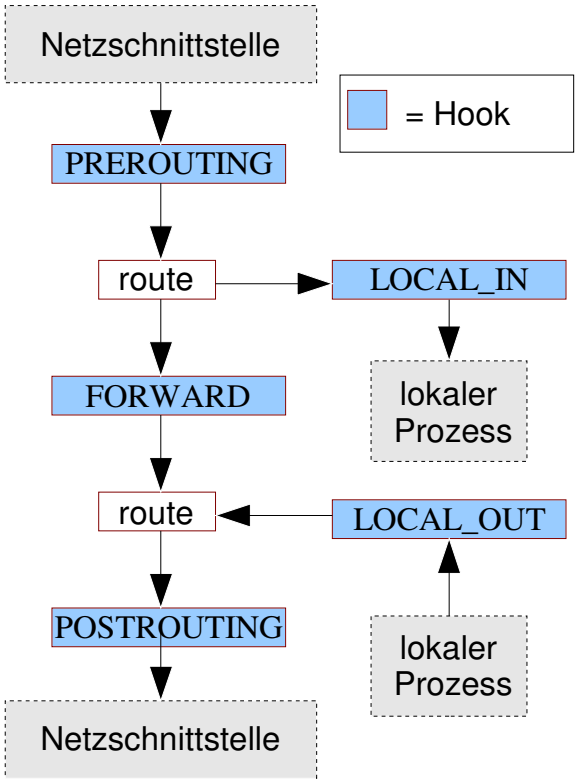

Abbildung 4 Netfilter-Filterprozess

Die vorliegende Implementation definiert eine Behandlungsroutine für Pakete, die am Netfilter- Hook NF_IP_POST_ROUTING, also kurz bevor sie wieder zum Versenden über die Netzwerkschnittstelle an die unteren Schichten weitergeleitet werden, ankommen.

Da das Abfangen von Netfilter-Hooks im hardwarenahen Kernelspace abläuft, Anwenderprogramme mit textbasierter oder grafischer Oberfläche jedoch im Userspace, wurde die Implementation des Emulators auf ein Kernelmodul und eine Konsolenanwendung aufgeteilt (siehe Abbildung 5).

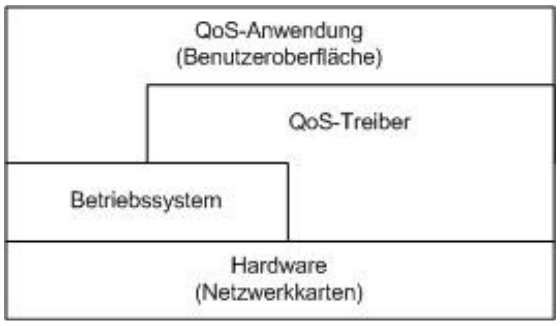

Abbildung 5 Architektur der Umsetzung

Das Kernelmodul erzeugt ein virtuelles, zeichenorientiertes Gerät und führt die eigentliche Emulation durch, indem es sich als Paketbehandlungsroutine am Hook *NF_IP_POST_ROUTING* registriert, die Pakete filtert und entsprechend den im Abschnitt 2.1 beschriebenen Algorithmen verzögert bzw. verwirft. Dazu werden für jede Netzschnittstelle zwei Warteschlangen eingerichtet, in denen eingehende Pakete eingereiht bzw. eingereihte Pakete verzögert werden. Darüber hinaus werden für jede Schnittstelle drei Threads gestartet:
1. Ein Arbeiter-Thread entnimmt die Pakete aus der Eingangswarteschlange, verwirft sie gegebenenfalls, setzt sie auf die Verzögerungswarteschlange und summiert die Größe der Pakete in einer globalen Variablen auf. Ist die eingestellte Bandbreite erreicht, pausiert der Thread, bis er wieder geweckt wird.
2. Der so genannte Clear-Thread wacht in einem vordefinierten Zeitintervall (zwischen 10 ms und 1 s) auf und setzt den aufsummierten Wert wieder auf 0.
3. Der Ausgabe-Thread entnimmt die Pakete der Verzögerungswarteschlange und vergleicht den Eingangszeitstempel mit der aktuellen Zeit:
   a) Ist die Differenz zwischen beiden größer gleich der eingestellten Verzögerung, wird das Paket ausgegeben
   b) Ist sie kleiner als die gewünschte Verzögerung wieder an den Anfang der Schlange gesetzt und der Thread pausiert maximal für die errechnete Differenz, minimal jedoch für 10 ms.

Die Konsolenanwendung greift über die *ioctl*-Schnittstelle auf das Kernelmodul zu, um mittels vordefinierter Befehle die Emulation zu starten, stoppen oder die Parameter beeinflussen zu können. Es besteht die Möglichkeit, auf einige fest kodierte Profile zurückzugreifen, Profile aus Dateien zu laden oder die Parameter der Emulation - mit gewissen Einschränkungen - frei zu wählen.

3 Erfahrungen aus der Anwendung

Der beschriebene Emulator wurde bereits für eine Reihe von Evaluationen im Rahmen von Forschungsprojekten, Diplomarbeiten sowie für Praktika eingesetzt. Hierbei hat sich gezeigt, dass der gewählte Ansatz für die Emulation einfacher Peer-to-Peer-Verbindungen durchaus tauglich ist. So wurden u. a. verschiedene Videokonferenzsysteme auf ihre Toleranz gegenüber langsamen bzw. fehlerbehafteten Verbindungen überprüft. Dabei zeigte sich, dass die verschiedenen Videokonferenzsysteme sehr unterschiedlich auf konstante Verzögerungen oder Paketverluste reagieren. Zum einen variieren Größe von Delay und Paketverlusten, bei denen eine Reaktion sichtbar wird, zwischen den Systemen mitunter, zum anderen reichen die Beobachtungen von kurzzeitigem Aussetzen bis zum kompletten Verbindungsabbruch. Auch die Audio- und Videoübertragungen zeigen unterschiedlich Sensibilitäten gegenüber Verschlechterungen der Dienstgüteeigenschaften.
Des weiteren wurde die im Rahmen einer Diplomarbeit erstellte H.323-Komponente auf ihre Verwendbarkeit über eine ISDN-Verbindung mit zwei B-Kanälen bei 128 kbit/s geprüft und mit einem Referenzsystem verglichen.
Als verbesserungswürdig hat sich vor allem die Benutzeroberfläche herausgestellt. Diese ist zum gegenwärtigen Zeitpunkt noch als textbasierte Konsolenanwendung realisiert, es ist jedoch geplant, diese durch ein einfaches Webinterface zur Einstellung der Emulationsparameter zu ersetzen und zudem die Visualisierung der aktuell anliegenden Parameter und Eigenschaften in Echtzeit hinzuzufügen.
Darüber hinaus ist für die Emulation einiger Profile eine bessere minimale Zeitauflösung als die momentan verfügbaren 10 ms nötig. Hierzu wird derzeit die Portierung des Emulators von der Linux-Kernel-Version 2.4 auf die Version 2.6 vorgenommen.

4 Zusammenfassung und Ausblick

Dieser Beitrag hat gezeigt, dass sich das vorgestellte Konzept sehr gut zur Emulation von Dienstgüte für IP- Netzwerke eignet, der Leistungsumfang des Emulators jedoch noch um einige Leistungsmerkmale wie z.B. eine präzisere Zeitauflösung sowie eine verbesserte Bedienung ergänzt werden kann.
Parallel zur Entwicklung des vorliegenden Emulators wurden Ansätze zur Netzwerkemulation in den LinuxKernel integriert, die mittlerweile als Netzwerkemulator unter dem Namen *NetEm* (siehe [8]) bekannt sind. Dieser setzt zwischen der IP-Schicht und den unteren Netzwerkschichten an und hat durch den Einsatz in der internationalen Linux-Anwender- und Entwicklergemeinde einen beachtlichen Funktionsumfang erreicht.
Der vorhandene Emulator soll nun so erweitert werden, dass er mögliche Erweiterungen auf Basis von NetEm unterstützt. Außerdem sollen für weitere Netze wie z.B. Mobile WiMax entsprechende Profile implementiert werden.

5 Literatur

[1] Steinmetz, R.: Multimedia-Technologie – Grundlagen, Komponenten und Systeme, Springer, 1999

[2] ISO/IEC IS 10746: The Reference Model of Open Distributed Processing (RM-ODP), ISO/IEC, 1996

[3] Detken, K.-O.: Echtzeitplattformen für das Internet, Addison-Wesley Verlag, 2002

[4] Freitag, B.: Entwicklung eines medienstrombezogenen QoS- Analysators und Simulators für ein drahtloses Teleservicesystem, Diplomarbeit für die Airbus Deutschland GmbH im Rahmen des Forschungsprojektes Multimedia Maintenance Assistance, Fachhochschule Osnabrück, 2005

[5] Paxson, V., Almes, G., Mahdavi, J. Mathis, M.: Framework for IP Performance Metrics, IETF RFC2330, 1998

[6] Westerkamp, C., Ramler, B., Wendler, R.: Nutzung heterogener Funknetze für mobilen Teleservice, Mobilfunk – Technologien und Anwendungen, Vorträge der 10. ITG-Fachtagung vom 1. bis 2. Juni 2005 in Osnabrück , VDE Verlag, 2005

[7] The Free Software Foundation: GNU General Public License, Version 2, Juni 1991, http://www.fsf.org/licensing/licenses/gpl.txt

[8] Hemminger, S.: Network Emulation with NetEm,