

# Staukontrolle durch Active Queue Management

Thomas Fischer und Dominik Billing

Betreuer: Martin Metzker

Seminar Kommunikationssysteme im Sommersemester 2014  
Lehrstuhl für Kommunikationssysteme und Systemprogrammierung  
Ludwig-Maximilians-Universität München

**Zusammenfassung**—Bei konventioneller Staukontrolle in Routern werden Pakete nicht gleichmäßig fallengelassen, wodurch es zu einem großen Overhead kommt, der durch Flaschenhälse im Internet noch verstärkt wird. Zusätzlich können Datenströme, die besonders viele Daten übertragen, nicht sinnvoll begrenzt werden. Wir werden als Lösung für das Problem der Staukontrolle im Internet Active Queue Management herausarbeiten. Active Queue Management versucht Staus frühzeitig und zuverlässig zu erkennen und durch Fallenlassen oder Markieren von Paketen (ECN – Explicit Congestion Notification) Staus zu verhindern. Dies wird von AQM-Algorithmen erreicht, indem die mittlere Pufferauslastung von Routern möglichst gering gehalten wird.

Um einen Überblick über aktuelle AQM-Algorithmen zu erhalten, werden wir die AQM-Methoden RED (Random Early Detection), BLUE und AVQ (Adaptive Virtual Queue) vorstellen und untereinander vergleichen. Das Prinzip von RED ist, Pakete bereits vor dem Überlaufen der Queue mit einer bestimmten Wahrscheinlichkeit fallen zu lassen, um Staus vor dem Entstehen zu verhindern. Die Wahrscheinlichkeiten berechnen sich dabei mittels der aktuellen Queue Auslastung. Bei BLUE werden die ankommenden Pakete auch mit einer Wahrscheinlichkeit schon vorher fallen gelassen, diese berechnet sich mittels der verlorenen Pakete. Bei AVQ wird die Kapazität einer virtuellen Queue als Entscheidung für das Fallenlassen herangezogen.

**Schlagwörter**—Staukontrolle, AQM, ECN, RED, BLUE, AVQ

## I. EINFÜHRUNG UND MOTIVATION

Die Ende-zu-Ende (E2E) Staukontrollmechanismen von TCP sind mittlerweile ein kritischer Faktor der Robustheit des Internets. Das Internet wächst unaufhaltsam weiter, es gibt keine eng verknüpfte Netzgemeinschaft und nicht jeder Endknoten verwendet die E2E Staukontrolle für bestmöglichen Datenfluss. Da auch Anwendungsentwickler sich nicht darum kümmern, E2E Staukontrolle in ihre Internet-Anwendungen zu integrieren, muss das Netz selbst seine Ressourcennutzung kontrollieren [1].

Mit der Entwicklung von immer leistungsfähigeren Prozessoren, Arbeitsspeichern und Festplatten ist die knappste Ressource in Netzen aktuell die Übertragungsrate. Das bedeutet, dass nicht alle eingehenden Daten verarbeitet und weiter versendet werden können. Wenn die Puffer eines Routers voll laufen, werden neu ankommende Pakete nach dem „Drop Tail“ Prinzip direkt abgewiesen (siehe Abbildung 1). Hier müssen in Situationen hohen Datenaufkommens Pakete mehrmals versandt werden, was zu Lasten der Netzgeschwindigkeit geht. Das Transportschicht Protokoll TCP (Transmission Control Protocol) erkennt genau dann einen Stau, wenn Pakete im Fluss fehlen, also vorher schon fallen gelassen wurden [2].

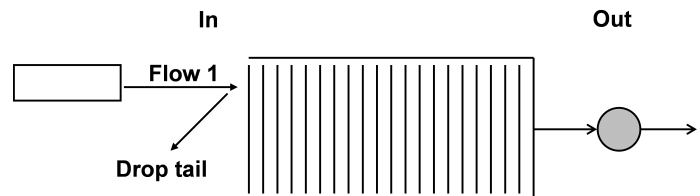


Abbildung 1. Das Prinzip von „Drop Tail“: Erreicht ein neues Paket ein System mit vollem Puffer, wird es fallengelassen [2].

Die Problemstellung ist es folglich Mechanismen zu finden, die Staus in E2E Verbindungen kontrollieren und frühzeitig identifizieren, damit Staus vermieden werden.

Um eine möglichst gute E2E Staukontrolle zu erreichen, soll die durchschnittliche Pufferausnutzung klein gehalten werden [3]. Die Herausforderung besteht darin, dass an den Flaschenhälsen der potenzielle Stau erkannt wird und das komplette Netz darauf reagiert, indem beispielsweise die Sendegeschwindigkeit des Ursprungs eines Datenstroms reduziert wird [2]. Interaktive Datenströme wie beispielsweise telnet, Web-Browsing und der Transfer von Audio- und Video-Material können sehr anfällig für Paketverluste oder hohe Latenzen sein, die auftreten, wenn Pakete erneut verschickt werden müssen [4].

Das folgende Kapitel II analysiert das Problem der Staukontrolle in Netzen und präsentiert Active Queue Management als Lösungsansatz für das Problem. Zusätzlich wird mit Explicit Congestion Notification (ECN) eine Alternative zum einfachen Fallenlassen von Paketen vorgestellt. Anschließend führt Kapitel III AQM ein und definiert Voraussetzungen eines funktionierenden Algorithmus. Die Active Queue Management Algorithmen RED, BLUE und AVQ werden in Kapitel IV vorgestellt und in Kapitel V miteinander verglichen. Im abschließenden Kapitel VI erfolgt eine Zusammenfassung der Ergebnisse zusammen mit einem Ausblick auf zukünftige Entwicklungen und Forschungen sowie andere Ansätze zur Staukontrolle.

## II. STAU-KONTROLLE IN NETZEN

Alle Staukontrollmechanismen müssen sich mit denen von TCP, als dem wichtigsten Transportschicht Protokoll im Internet, gemessen an den Datenflüssen, vergleichen [5]. Neben [1] wird auch von Morris postuliert, dass die Effektivität von TCP mit zunehmender Anzahl an konkurrierender Datenströme nachlässt [6]. Diese Probleme sind auch im Internet spürbar

und es zeigt sich, dass das Internet einen gravierenden Performanceverlust dadurch erfährt. Die einfachste Lösung für diese Probleme wäre die radikale Vergrößerung von Routerpuffern, zusammen mit einer Begrenzung der Anzahl der Pakete jedes einzelnen Datenstroms individuell. Da dies aber nicht ausführbar ist, müssen Staukontrollalgorithmen implementiert werden, mit den folgenden Hauptzielen [6]:

- Es soll eine hohe Ausnutzung von Flaschenhälsen wie Routern erreicht werden.
- Der Überlauf von Flaschenhälsen und damit eine zeitliche Verzögerung durch Staus sowie ein hoher Paketverlust soll verhindert werden.
- Die zur Verfügung stehende Übertragungsrate soll gleichmäßig zwischen konkurrierenden Datenströmen aufgeteilt werden.

Um diese gleichmäßige Verteilung zu erreichen, ist TCP standardmäßig mit dem „Drop Tail“ Prinzip nicht geeignet. Deshalb müssen Methoden entwickelt werden, die Puffer in Routern anders abarbeiten und dennoch einfach zu verwenden sind [7].

Die Internet Architektur nutzt größtenteils zum Weiterleiten das verbindungslose IP-Protokoll, auf dem dann verbindungslose (UDP) oder verbindungsorientierte (TCP) Dienste der Transportschicht aufsetzen. Diese Architektur hat vor allem aufgrund des verbindungslosen Designs, der Flexibilität und der Robustheit viele Vorteile, allerdings auch folgende Nachteile [8]:

- Sorgfältiges Design ist von Nöten, um bei hoher Last einen guten Dienst zu leisten.
- Mangelnde Aufmerksamkeit auf die Dynamik der Paketweiterleitung kann dazu führen, dass Dienste nicht mehr korrekt funktionieren und dass es schlimmstenfalls zu einem sogenannten „internet meltdown“ kommt.

Das Phänomen eines solchen Internet-Zusammenbruchs wurde während der ersten Wachstumsphase des Internets in den 1980er Jahren festgestellt und wird „congestion collapse“ [9] genannt. Bereits 1986 wurden von Jacobson entwickelte Stauverhinderungsmechanismen für Hosts entwickelt, die auch aktuell noch einen „congestion collapse“ verhindern [8].

Da das Internet seit dieser Zeit immer weiter wächst, wurde es offensichtlich, dass TCP Stauverhinderungsmechanismen [10] nicht unter allen Umständen ausreichend gute Dienste leisten. Das Hauptproblem liegt darin, dass von den Enden der Netze nur bedingt Kontrolle ausgeübt werden kann. Deshalb müssen auch in Routern Mechanismen angewendet werden, welche die Stauverhinderungsmechanismen der Endpunkte ergänzen. Hierbei muss man zwischen den folgenden zwei Klassen unterscheiden [8]:

**Queue Management** Algorithmen verwalten die Länge von Paket-Puffern durch Fallenlassen von Paketen wenn nötig oder angemessen.

**Scheduling** Algorithmen legen fest, welche Pakete als nächstes gesendet werden sollen und können primär dafür genutzt werden, die Zuweisung von Übertragungsraten zwischen den Datenströmen zu verwalten.

Ein guter Algorithmus vereint die Vorteile beider Arten in sich, dazu mehr in Kapitel III.

Nach Jain [11] gibt es zwei Gründe, warum das Problem der Staukontrolle in Netzen sehr schwierig ist. Erstens gibt es Voraussetzungen für Staukontrollschemas, die es schwierig machen eine zufriedenstellende Lösung zu finden. Zweitens gibt es unzählige Netzregeln, die das Design eines Stauschemas beeinflussen. Das sind die Gründe, warum ein Schema, das für ein Netz entwickelt wurde, in einem anderen Netz nicht funktioniert. Grundbedingungen für Schemata zur Staukontrolle sind:

- Ein Schema muss einen kleiner Overhead haben.
- Alle Datenströme müssen gleich behandelt werden.
- Es muss schnell auf andere Situationen reagiert werden können.
- Es muss in schlechten Umgebungen funktionsfähig sein.
- Es muss für alle Benutzer optimal sein.

Um E2E Staukontrolle in Routern zu betreiben, muss nicht nur jeder einzelne Router auf sich allein gestellt seinen Puffer überwachen, sondern auch an die nächsten Router Informationen weiterleiten. Für diese Benachrichtigung könnten zwei reservierte Bits im IP-Header genutzt werden [2]. Mittels „Explicit Congestion Notification“ (ECN) sollen Pakete mit der Staubenachrichtigung weiterversandt werden im Gegensatz zum einfachen Fallenlassen der Pakete. Auf diese Art und Weise werden vorangehende Router darüber informiert, dass es zu einem Stau gekommen ist und möglicherweise einzelne Pakete doppelt versandt werden müssen oder die Geschwindigkeit gedrosselt werden sollte. Router sind in der Vermittlungsschicht angesiedelt; es wird hier in der Regel das IP Protokoll verwendet, bei dem die IP-Adresse das Ziel eindeutig angibt. Im Gegensatz dazu befinden sich Switches in der Sicherungsschicht und leiten Pakete beispielsweise anhand der MAC-Adresse weiter. In kleinen Netzen ist es prinzipiell kein Problem Pakete einfach fallen zu lassen. Da im Internet E2E Verbindungen über sehr viele Router gehen können, kann das Fallenlassen von Paketen durchaus problematisch werden. ECN ist ein zuverlässiger Mechanismus, beinhaltet allerdings keine Methoden zur Erkennung von Staus [12]. Da es nicht vorhersehbar ist, wie hoch der Datenverkehr zukünftig sein wird, besteht das wirkliche Problem darin, Staus frühzeitig und zuverlässig zu erkennen [13].

Als Lösung für das Problem der Staukontrolle gehen wir auf „Active Queue Management“ (AQM) ein. AQM hat das Ziel Staus in Netzen rechtzeitig zu entdecken, bevor die Routerpuffer volllaufen. Um den Quellen von Datenströmen zu signalisieren, dass es zu Problemen gekommen ist oder kommen wird, gibt es im Prinzip die beiden Möglichkeiten Pakete fallen zu lassen oder diese zu markieren. Die erste Strategie erfordert es, dass die Endpunkte kooperieren und erzeugt durch erneuten Versand der Pakete einen Overhead, der bezüglich der verbrauchten Übertragungsrate messbar ist. Andererseits müssen Router auf markierte Pakete reagieren, als wären diese fallengelassen worden. Es wird hier allerdings kein weiterer Overhead erzeugt, da die Pakete nicht erneut gesendet werden müssen. AQM-Algorithmen können zusätzlich die Übertragungsrate von besonders gierigen Datenströmen reduzieren, indem deren Pakete häufiger fallengelassen oder markiert werden als andere. Einen wirklichen Unterschied

machen AQM-Algorithmen allerdings nur, wenn sie wirklich flächendeckend und der Situation entsprechend eingesetzt werden [2].

### III. DEFINITION UND ANWENDUNG VON ACTIVE QUEUE MANAGEMENT

TCP ist ein E2E Protokoll, das zusammen mit dem Internet Protokoll (IP) genutzt wird, um Daten in Form von Paketen zwischen Computern über das Internet zu übertragen. Während IP für die Vermittlung der Daten sorgt, gehören Datensicherheit, Flusssteuerung und das Ergreifen von Maßnahmen bei Datenverlust zu den Aufgaben von TCP. TCP implementiert einen Algorithmus zur Flusssteuerung, der „Sliding Window“ genannt wird. Das „Window“ ist die Anzahl an Bytes, die gesendet werden können ohne auf Empfangsbestätigung warten zu müssen. Der Algorithmus besteht aus folgenden Schritten [14], [15]:

- 1) Alle Bytes eines Windows verschicken.
- 2) Auf die Empfangsbestätigung warten (es können mehrere Pakete auf einmal zurück gemeldet werden).
- 3) Das Window bis an das Ende des lückenlos bestätigten Bytestroms verschieben.
- 4) Wenn die Empfangsbestätigung eines Pakets nicht in einer angegebenen Zeit erfolgt, dann wird das Paket neu verschickt.

TCP nimmt hier keine Rücksicht auf Staus im Netz. Wenn kontinuierlich immer weiter Pakete an das Netz versandt werden, ohne dass diese vom Netz aufgenommen werden können, führt dies zu einem Stau, der sich eventuell immer weiter aufbaut. An genau dieser Stelle muss ein Active Queue Management Algorithmus ansetzen und die Staukontrolle übernehmen [16].

Alle Internetrouter besitzen Puffer, um auf Verzögerungen im Fluss kurzfristig reagieren zu können und Daten eine kurze Zeit vorzuhalten. Die Größe von Routerpuffern ist durch Hardware festgelegt, orientiert sich allerdings an der Dynamik des Staukontrollalgorithmus von TCP. Genauer gesagt ist es das Ziel einen Puffer nie leer laufen zu lassen. Die Größe eines Puffers muss nach dieser Definition mit wachsender Verbindungsgeschwindigkeit ansteigen. Das Problem hierbei ist, dass auf diese Art große und langsame Speicher verwendet werden müssten. Durch die Anwendung von besseren Staukontrollalgorithmen wie AQM können diese kostspieligen Puffer klein bleiben [17].

Die Definition von Active Queue Management lautet wie folgt:

**Active Queue Management (AQM)** ist das aktive Neusortieren oder Fallenlassen von Paketen innerhalb eines Puffers. Damit sind AQM Algorithmen eine Mischung aus Scheduling und Queue Management Algorithmen, mit folgenden Zielen [18], [19]:

- Es sollen möglichst wenig Pakete fallengelassen werden.
- Daten aus einfachen Datenquellen wie beispielsweise telnet sollen mit einer sehr kurzen Verzögerung behandelt werden.
- Die Übertragungsrate soll gleichbehandelnd zwischen den unterschiedlichen Flüssen aufgeteilt werden.

- Staus sollen frühzeitig erkannt werden.
- Flüsse aus nicht responsiven Quellen sollen derart behandelt werden, dass responsive Flüsse davon nicht beeinträchtigt werden.
- Der Algorithmus soll einfach zu implementieren sein und schnell reagieren.

Das ist nur möglich, wenn sich alle Datenquellen untereinander derart koordinieren, dass die Queue-Größe unter Kontrolle gehalten wird. Das Problem hierbei besteht darin, dass Übertragungsrate, Schnelligkeit und Pufferplatz unabhängig von einander zugeordnet werden müssen[20].

Selbst wenn AQM Mechanismen verwendet werden, ist es oft sehr sinnvoll zusätzlich auch ECN zu verwenden, um einen unnötigen Overhead zu vermeiden. Bei ECN wird durch das Markieren eines Pakets die Kommunikation zwischen einzelnen Routern im Netz ermöglicht. Zusammen haben AQM und ECN das Potenzial den Effekt von Verlusten in latenzsensitiven Flüssen zu reduzieren [4]. Aus diesem Grund hängt der Erfolg jedes AQM Mechanismus stark damit zusammen, ob er mit ECN verbunden werden kann und wie gut diese Verbindung erreicht wird [21].

Im Laufe der Zeit ist sehr viel Aufwand in unterschiedliche AQM-Methoden und deren Feintuning gesteckt worden. Viele dieser Arbeiten basieren auf Heuristiken und Simulationen und nicht auf einem systematischen Ansatz. Ihr gemeinsames Problem ist, dass jede vorgeschlagene Konfiguration nur für ein spezielles Datenverkehrsaufkommen geeignet ist, aber nachteilige Effekte aufweist, wenn sie in einer anderen Umgebung angewendet wird. Entsprechend müssen AQM-Methoden entwickelt und ihre Parameter so eingestellt werden, dass sie in einer Vielzahl von unterschiedlichen Umgebungen einsetzbar sind und zumindest keine schlechteren Ergebnisse liefern als TCP selbst [22]. Es existieren sehr viele unterschiedliche AQM-Methoden, die komplett unterschiedliche Ansätze verfolgen. Manche Algorithmen berechnen beispielsweise Wahrscheinlichkeiten, um Pakete fallen zu lassen, andere nutzen Verfahren aus der mathematischen Optimierung aus, um die bestmögliche Staukontrolle zu erreichen. Alle Algorithmen unterscheiden sich in der Anzahl an möglichen Einsatzgebieten, Komplexität und Qualität [21].

Im nächsten Kapitel stellen wir drei der bekanntesten Algorithmen vor.

### IV. DREI BEISPIELE FÜR ACTIVE QUEUE MANAGEMENT ALGORITHMEN

Seitdem die ersten Ideen für Active Queue Management vorgestellt wurden und der Einführung von ECN in TCP/IP wurden viele verschiedene AQM Algorithmen entwickelt. Es würde weit über den Rahmen dieser Arbeit hinausgehen, sie alle zu erwähnen, weshalb hier die gängigsten drei RED, BLUE und AVQ vorgestellt werden. Es wurden diese drei Algorithmen ausgewählt, da zu diesen sehr viele Arbeiten existieren und sie bei vielen anderen Algorithmen als Referenz dienen.

#### A. RED: Random Early Detection

Der erste Algorithmus, der präsentiert wird, ist „Random Early Detection“ (RED). Er war einer der ersten AQM Algo-

rithmen und viele andere Arbeiten entwickelten diesen weiter, z.B. in [23] oder [24]. Algorithmen, die anders ablaufen, werden oft mit RED verglichen.

Floyd und Van Jacobson haben RED 1993 vorgestellt [25]. In ihrer Arbeit wird die Funktionsweise des Algorithmus dargestellt. Um die Sender über einen Stau zu informieren kann RED entweder Pakete fallen lassen oder das ECN-Bit im Header setzen, je nachdem ob der Router ECN-fähig ist. Im Folgenden werden wir nur die Möglichkeit des Markierens (ECN-Bit Setzen) betrachten, was auf den eigentlichen Algorithmus keinerlei Auswirkungen hat.

Das Prinzip von RED ist es Pakete mit einer Wahrscheinlichkeit zu markieren, die sich proportional zum Anteil der Übertragungsrate verhält, den diese Verbindung belegt. Auf diese Weise versucht der Algorithmus die Ressourcenzuteilung fair zu machen.

Als Messgröße wird die durchschnittliche Queue Länge benutzt. Die durchschnittliche Länge  $Q_{avg}$  wird für jedes eintreffende Paket mittels der aktuellen Länge der Queue  $q$  und dem Gewicht der Queue  $w_q$  folgendermaßen neu berechnet:

$$Q_{avg} = (1 - w_q)Q_{avg} + w_q q$$

Dieser Wert wird mit zwei Parametern verglichen, der minimalen Queue Länge  $Q_{min}$  und der maximalen Queue Länge  $Q_{max}$ . Ist  $Q_{min} > Q_{avg}$ , so wird nichts unternommen. Wird aber  $Q_{min} < Q_{avg} < Q_{max}$ , so wird das Paket mit einer Markierungswahrscheinlichkeit  $p_a$  markiert, und sobald  $Q_{avg} > Q_{max}$  wird jedes Paket markiert.

Für die Berechnung der finalen Markierungswahrscheinlichkeit  $p_a$  wird die Markierungswahrscheinlichkeit  $p_b$  benötigt. Diese berechnet sich aus der minimalen und maximalen Queue Länge  $Q_{min}$  und  $Q_{max}$ , der Durchschnittslänge  $Q_{avg}$  und dem Maximum für  $p_b$ ,  $max_b$ , wie folgt:

$$p_b = max_b \frac{Q_{avg} - Q_{min}}{Q_{max} - Q_{min}}$$

$p_b$  steigt folglich linear von 0 bis zum Wert  $max_b$  an. Die finale Markierungswahrscheinlichkeit wird mittels  $p_b$  und eines Zählers  $z$  berechnet:

$$p_a = \frac{p_b}{1 - z p_b}$$

Der Zähler wird für jedes einkommende Paket inkrementiert. Ein Paket wird mit der Wahrscheinlichkeit  $p_a$  markiert. Sobald ein Paket tatsächlich markiert wird, wird der Zähler  $z$  wieder zurück auf 0 gesetzt. Mithilfe des Zählers steigt die Wahrscheinlichkeit somit für jedes weitere Paket an.

Damit ergibt sich als Algorithmus für RED:

Neben der Queue Länge in Paketen könnte RED auch die Bytelänge und somit die Anzahl an Bytes eines Pakets zur Bewertung heranziehen. Diese Information wird dann in die Markierungswahrscheinlichkeit  $p_b$  mit einbezogen. Nach der Berechnung ändert sich der Wert dann in

$$p_b = p_b \frac{\text{Paketbytes}}{\text{maximale Paketbytes}}$$

Dadurch werden große Pakete mit einer höheren Wahrscheinlichkeit markiert als kleine Pakete. An der Beschreibung des Algorithmus wird ersichtlich, dass RED viele Parameter benötigt, die vorab festgelegt werden müssen. Für

```

for jedes ankommende Paket do
  Berechne  $Q_{avg}$  if  $Q_{min} < Q_{avg} < Q_{max}$  then
    Berechne die Wahrscheinlichkeit  $p_a = \frac{p_b}{1 - z p_b}$  ;
    Mit der Wahrscheinlichkeit  $p_a$ : Markiere das ankommende Paket ;
  else
    if  $Q_{max} < Q_{avg}$  then
      Markiere das ankommende Paket ;
    end
  end
end

```

#### Algorithmus 1 : RED

jede Situation müssen die richtigen Werte gefunden werden, damit RED gute Ergebnisse liefert. Das stellt ein nicht zu vernachlässigendes Problem des Algorithmus dar. Mehr dazu in Kapitel V.

#### B. BLUE

Der BLUE Algorithmus wurde 1999 von Feng et.al. an der University of Michigan in Zusammenarbeit mit IBM vorgestellt [26]. BLUE wurde entwickelt, um einige Schwachstellen von RED zu verbessern, ist aber ein völlig neuer Ansatz. RED verlässt sich auf die Queue Länge, um Staus zu erkennen, und benötigt viele Parameter, die konfiguriert werden müssen. Die Autoren von BLUE argumentieren, dass RED nur wenn diese richtig konfiguriert sind und wenn ausreichend Pufferplatz zur Verfügung steht optimal läuft. BLUE dagegen verlässt sich auf den Paketverlust und die Verbindungsauslastung um seine Markierungswahrscheinlichkeit zu berechnen. Genauso wie RED kann BLUE dann entweder Pakete fallen lassen oder mittels ECN markieren.

Der BLUE Algorithmus kennt nur eine Markierungswahrscheinlichkeit  $p_m$ ; jedes eintreffende Paket wird mit dieser Wahrscheinlichkeit markiert. Die Entscheidung für die Erhöhung oder Erniedrigung dieser Wahrscheinlichkeit wird auf Basis der verlorenen Pakete beziehungsweise auf Basis der ungenutzten Verbindungen getroffen: Erhält der Router die Information, dass ein Paket verloren gegangen ist, wird  $p_m$  um den Wert  $d_1$  erhöht. Erkennt er eine ungenutzte Verbindung, wird  $p_m$  um  $d_2$  reduziert. Ein weiterer Parameter ist hierbei noch wichtig: die *freeze\_time*. Damit das Netz und die Sender Zeit haben, auf die Aktion des Routers zu reagieren, muss mindestens dieses Zeitintervall vergangen sein, bis die Markierungswahrscheinlichkeit wieder geändert wird. Formal läuft ein Schritt des Algorithmus folgendermaßen ab:

Die beiden Parameter  $d_1$  und  $d_2$  geben an, um wie viel  $p_m$  zu erhöhen beziehungsweise zu reduzieren ist.  $d_1$  sollte deutlich größer sein als  $d_2$ , da BLUE somit auf Staus sehr viel schneller reagieren kann. Die Autoren geben weiterhin an, dass in ihren Experimenten die *freeze\_time* konstant gehalten wurde. Sie sagen aber auch, dass in einem Netz dieser Parameter zufällig für jeden Router gewählt werden sollte, um globale Synchronisation zu vermeiden. Dieser Algorithmus passt sich somit selbstständig an den aktuellen Bedarf des Netzes an und benötigt keine Router-abhängigen Parameter zur Konfiguration.

```

for jedes ankommende Paket do
  if Paketverlust  $\wedge$  (now - last_update) < freeze_time
  then
     $p_m = p_m + d_1$  ;
    last_update = now ;
  end
  if Verbindung frei  $\wedge$  (now - last_update) <
  freeze_time then
     $p_m = p_m - d_2$  ;
    last_update = now ;
  end
end

```

**Algorithmus 2 : BLUE**

### C. AVQ: Adaptive Virtual Queue

Ein weiterer AQM-Algorithmus, der eine andere Idee verfolgt, ist der Adaptive Virtual Queue (AVQ) Algorithmus. Er wurde 2001 von Kunniyur und Srikant vorgestellt [27]. Wie bereits der Name andeutet, verwendet der Algorithmus eine virtuelle Queue. Für die Markierung beziehungsweise das Fallenlassen von Paketen wird die Kapazität dieser virtuellen Queue und keine Markierungswahrscheinlichkeit zu Rate gezogen.

Bei AVQ verwaltet der Router neben der echten Queue eine virtuelle Queue mit der Kapazität  $C_v \leq C$ , wobei  $C$  die Kapazität der Verbindung und  $C_v$  die der virtuellen ist. Zu Beginn ist  $C_v = C$ . Bei jedem ankommenden Paket wird überprüft, ob der Puffer der virtuellen Queue das Paket aufnehmen könnte. Ist dem so, wird das echte Paket in die tatsächliche Queue eingereiht, ansonsten wird es markiert oder fallen gelassen. Die Kapazität der virtuellen Queue wird ebenfalls bei jedem ankommenden Paket angepasst gemäß der Differentialgleichung (siehe auch Abbildung 2)

$$\dot{C}_v = \alpha(\gamma C - \lambda)$$

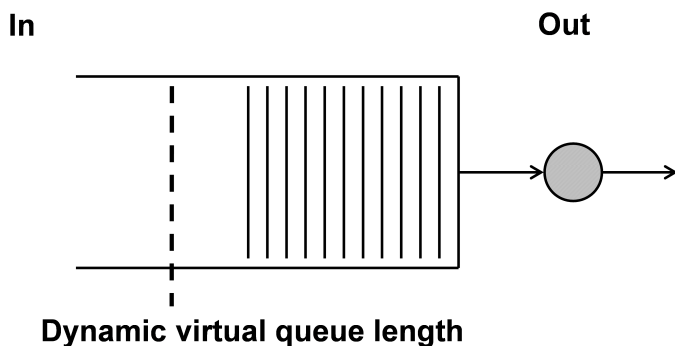


Abbildung 2. Das Prinzip von AVQ: Die Größe der virtuellen Queue wird dynamisch angepasst, um bessere Leistungsgrenzen zu erhalten [2].

Hierbei ist  $\alpha$  ein Glättungsparameter,  $\gamma$  die angestrebte Auslastung der Verbindung und  $\lambda$  die Ankunftsrate der Verbindung. Das Markieren passiert auf diese Weise aggressiver, also häufiger, wenn die Verbindung ihre gewünschte Übertragungsrate überschreitet und weniger aggressiv, wenn nicht.

Es ist klar, dass in der virtuellen Queue keine Pakete eingereiht werden müssen, lediglich die Länge, also die Kapazität der virtuellen Queue muss ermittelt werden. Die Autoren geben in ihrer Arbeit auch an, wie dieses Verfahren implementiert wird:

```

for jedes ankommende Paket do
  if  $VQ = \max(VQ - C_v(t - s), 0)$  then
    Paket markieren ;
  else
     $VQ = VQ + b$  ;
  end
   $C_v = \max(\min(C_v + \alpha \cdot \gamma \cdot C(t - s), C) - \alpha \cdot b, 0)$  ;
   $s = t$  ;
end

```

**Algorithmus 3 : AVQ**

Hierbei ist  $B$  die Puffergröße in Bytes,  $s$  die Ankunftszeit des letzten Pakets,  $t$  die aktuelle Zeit,  $b$  die Paketgröße des aktuellen Pakets in Bytes und  $VQ$  die Anzahl an Bytes, die aktuell in der virtuellen Queue sind.

Die Autoren erläutern, dass die algorithmische Komplexität von AVQ in etwa der von RED entspricht. Anstelle der Länge der Queue wird bei AVQ jedoch die Ausnutzung der Queue als Entscheidungskriterium für das Markieren von Paketen verwendet. Für die Verwendung von AVQ müssen der Glättungsparameter  $\alpha$  und die gewünschte Ausnutzung  $\gamma$  vorab angegeben werden. Diese dienen der Stabilität des Verfahrens. Weiter geben die Autoren an, dass  $\gamma$  es den ISP's erlaubt einen Ausgleich zwischen hoher Übertragungsratenausnutzung und kleinen Puffern vorzunehmen. Eine Regel, wie diese Parameter gesetzt werden sollten, befindet sich in der Originalarbeit in [27].

## V. VERGLEICH DER VORGESTELLTEN ALGORITHMEN

Nachdem wir im vorigen Kapitel IV die AQM Algorithmen RED, BLUE und AVQ vorgestellt haben, werden wir anschließend diese untereinander vergleichen.

In der zur Verfügung stehenden Literatur lies sich kein Vergleich zwischen den Algorithmen BLUE und AVQ finden. Deshalb kann auch an dieser Stelle kein solcher Vergleich vorgenommen werden.

### A. Vergleich BLUE und RED

In ihrer Arbeit zeigen Feng et al anhand einiger Versuche, dass BLUE im Vergleich zu RED deutlich weniger Pakete fallen lässt [26]. Für ihren Versuch haben sie ein Netz mithilfe des LBNL Network Simulators [28] simuliert. Für alle Quellen wurde ECN aktiviert, was bedeutet, dass jedes verlorene Paket einen Pufferüberlauf der Queue darstellt. Die Sender wurden zufällig innerhalb der ersten Sekunde gestartet. Die Paketverluste wurden nach 100 Sekunden Simulation und weiteren 100 Sekunden, also insgesamt 200 Sekunden, für die gesamte Simulationszeit gemessen. Die Parameter für RED wurden experimentell bestimmt,  $Q_{min}$  betrug immer 20% der Queuelänge,  $Q_{max}$  80%. Für BLUE wurde  $d_1$  um eine Größenordnung höher als  $d_2$  gesetzt. Insgesamt wurden vier

Konfigurationen für RED und vier Konfigurationen für den BLUE Algorithmus angewandt und die Ergebnisse verglichen.

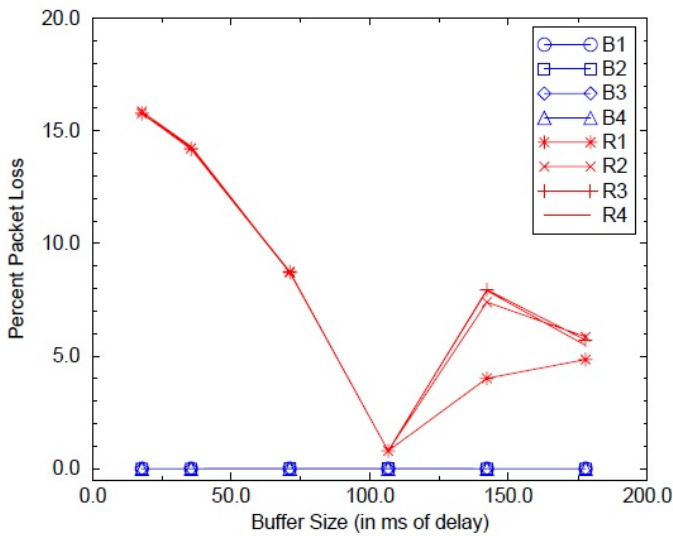


Abbildung 3. Vergleich der Verlustraten von RED und BLUE bei 1000 Quellen [26].

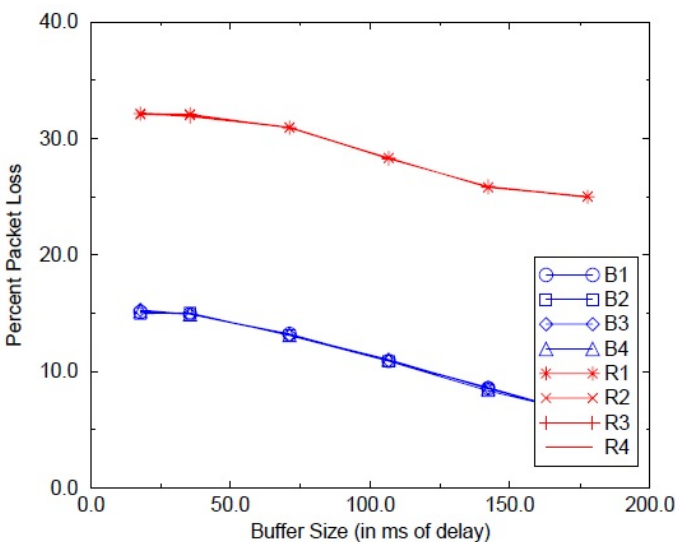


Abbildung 4. Vergleich der Verlustraten von RED und BLUE bei 4000 Quellen [26].

Zwischen den Ergebnissen der einzelnen Konfigurationen sowohl bei RED als auch bei BLUE gab es nur minimale Unterschiede. Die Queuelänge für eine Engstelle des Netzes wurde von 100KB bis 1000KB gesetzt, was einer Verzögerung zwischen 17,8ms und 178ms entspricht. Bei 1000 Quellen, die gleichzeitig senden, war sowohl bei RED als auch bei BLUE die Verbindung für alle Queueängen zu 100% ausgelastet. Die Paketverlustraten sind in Abbildung 3 dargestellt. Bei BLUE betrug sie für alle Verzögerungen 0%. Der RED Algorithmus dagegen zeigt bei niedrigen Verzögerungen eine Paketverlustrate von bis zu über 15%. Diese fällt mit größer werdenden Puffern auf knapp über 0% bei einer Verzögerung von etwa 100ms, steigt dann aber wieder an auf etwa 5%. Für

das gleiche Experiment mit 4000 Quellen war die Auslastung ebenfalls bei beiden stets bei 100%. Die Paketverlustrate von RED lag diesmal bei kleinen Verzögerungen bei über 30%, was mit zunehmenden Queueängen auf etwa 25% abfiel. Bei BLUE lag die Verlustrate dagegen zu Beginn bei etwa 15% und sank auf unter 10% mit größer werdenden Queues (siehe Abbildung 4).

### B. Vergleich AVQ und RED

Analog zu BLUE hat auch Kunniyar in seiner Arbeit zu AVQ Vergleiche des Verfahrens zu anderen AQM Algorithmen gemacht [27]. Neben RED wurde der Algorithmus auch mit Random Early Marking (REM), dem PI Controller und GKVQ verglichen. Diese werden hier jedoch vernachlässigt, da auch die Algorithmen nicht präsentiert worden sind, allerdings wurde unter den vorgegebenen Randparametern mit AVQ ein besseres Ergebnis als mit allen anderen Algorithmen erzielt.

Die beschriebenen Versuche wurden mit dem Simulator *ns-2* durchgeführt. Der Parameter  $\gamma$ , die angestrebte Auslastung, wurde für AVQ auf den Wert 0,98 gesetzt; der Glättungsparameter  $\alpha$  wurde auf 0,15 gesetzt. Die Konfiguration für RED wurde gemäß den Empfehlungen aus [29] vorgenommen. Die Queuelänge an der engsten und somit relevanten Stelle der Verbindung betrug 100 Pakete bei einer Paketgröße von 1000 bytes. Für das erste Experiment wurde bei den TCP Verbindungen ECN aktiviert. Jedes verlorene Paket ist folglich ein Zeichen des Pufferüberlaufs. Die Grenzwerte für RED wurden auf  $Q_{min} = 0,37$  und  $Q_{max} = 0,75$  gesetzt. Die durchschnittliche Verzögerung betrug zwischen 30ms und 60ms. Beim Experiment wurde die Paketverlustrate sowie die Auslastung der Verbindung bei unterschiedlicher Anzahl von FTP Verbindungen gemessen, die Anzahl der Verbindungen lag zwischen 20 und 180.

Die Auslastung der Verbindung lag bei RED bei 20 FTP Verbindungen bei knapp unter 90%. Mit steigender Anzahl an Verbindungen sank diese leicht ab, bewegte sich aber stets zwischen 85% und 90%. Für AVQ lag die Auslastung bei 20 Verbindungen bei 95%. Diese stieg kontinuierlich mit steigenden Verbindungen an und lag bei 180 Verbindungen etwa bei 98%, der vorab festgelegten angestrebten Auslastung. Die Paketverluste von RED lagen bei 20 Verbindungen etwa bei 0. Mit steigenden FTP Verbindungen stiegen diese in etwa linear auf über 6.000 verlorene Pakete bei 180 Verbindungen (siehe Abbildung 5, schwarz). Bei AVQ lagen die Paketverluste für jede Anzahl an Verbindungen bei 0 (siehe Abbildung 5, blau). Der AVQ Algorithmus zeigt für diesen Aufbau eine deutlich besser Leistung mit höherer Auslastung und ohne Paketverluste.

In einem zweiten Experiment wurde ECN deaktiviert. Das heißt, dass sowohl RED als auch AVQ Pakete fallen lassen, um auf einen Stau hinzuweisen. Der Versuchsaufbau ist gleich dem vorherigen mit der Ausnahme, dass für AVQ die angestrebte Auslastung  $\gamma$  auf den Wert 1 gesetzt wurde, die Adaptive Virtual Queue also voll genutzt werden soll. Die Anzahl an FTP Verbindungen wurde fest auf 40 gesetzt; dafür wurden TCP short-flows mit 20 Paketen verwendet, deren Ankunftszeit an der kritischen Verbindungsstelle langsam gesteigert wurde.



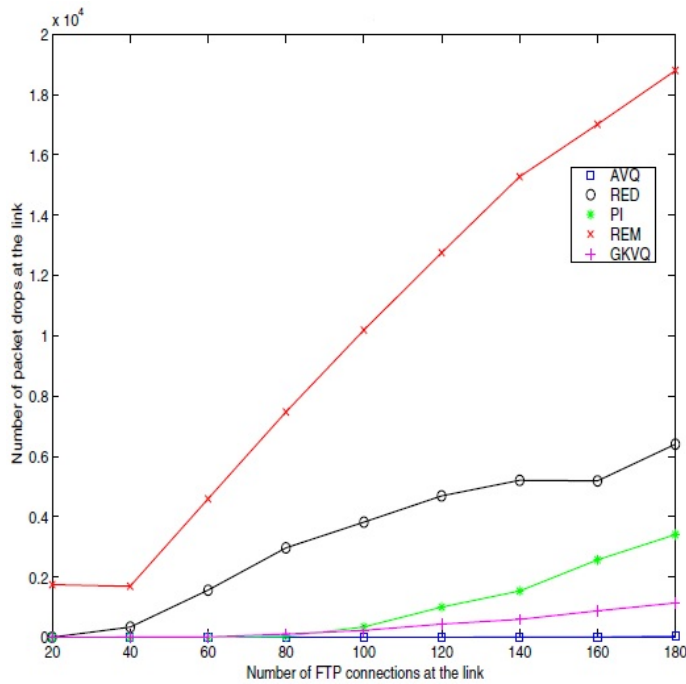


Abbildung 5. Absolute Anzahl an Paketverlusten von verschiedenen AQM Verfahren bei steigender Anzahl an FTP-Verbindungen [27].

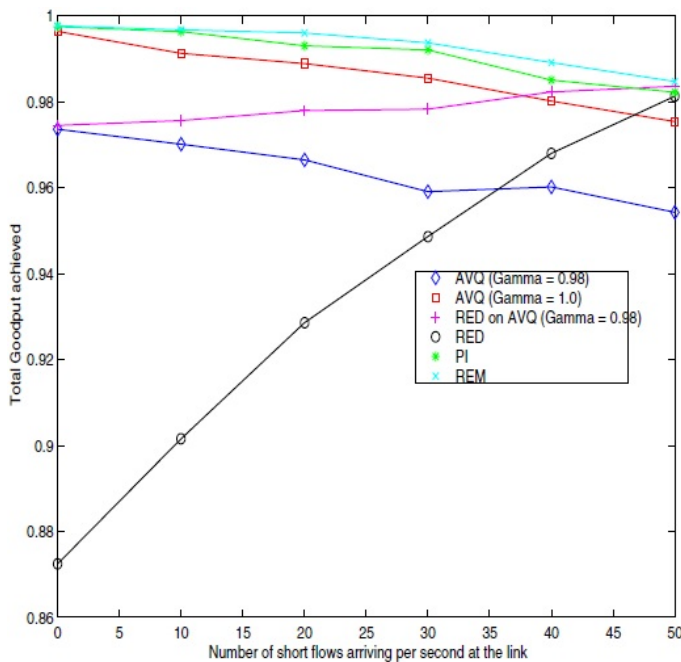


Abbildung 6. Anteil korrekt zugestellter Pakete von verschiedenen AQM Verfahren bei steigender Anzahl an short flows [27].

Die Auslastung der Verbindung für AVQ lag konstant bei den angestrebten 100%. Bei RED lag diese bei 10 ankommenden short-flows pro Sekunde bei etwa 94% und stieg auf über 99% bei 50 ankommenden short-flows pro Sekunde. Der Anteil an korrekt zugestellten Paketen, ohne diese einmal fallen zu lassen, lag bei AVQ bei wenigen short-flows bei knapp unter 100% und sank bis zu knapp unter 98% bei 50 ankommenden short-flows pro Sekunde (siehe Abbildung 6, rot). Für RED

lag diese Quote anfangs nur bei 87%, stieg dann aber stark an auf 98% bei 50 short-flows pro Sekunde und lag damit höher als bei AVQ (siehe Abbildung 6, schwarz).

### C. Zusammenfassung

Zu Beginn dieser Arbeit wurde die Notwendigkeit für AQM Algorithmen erläutert. Sie sind notwendig, um aktiv einen Pufferüberlauf der Router in großen Netzen zu verhindern. RED war einer der ersten Algorithmen, der dies zum Ziel hatte. Bereits 1998 wurde von der Network Working Group in RFC 2309 empfohlen, einen AQM Mechanismus zu implementieren, wobei explizit RED als Verfahren vorgeschlagen wurde [8]. Ebenfalls wurde empfohlen, weiter Forschung in diesem Bereich zu betreiben. Hervorzuheben ist auch, dass die Empfehlung für RED vor der Empfehlung für ECN ausgegeben wurde, welche 1999 in RFC 2481 gegeben wurde [30].

Aus dieser Forschung sind zahlreiche weitere AQM Algorithmen hervorgegangen. Hier wurden davon die Verfahren BLUE und AVQ vorgestellt und mit RED verglichen. Die Autoren von BLUE zeigen mit ihren Versuchen, dass BLUE eine deutlich geringere Paketverlustrate als RED aufweist und besonders mit kleinen Puffern noch eine gute Performance zeigt. In den Experimenten zur Leistung von AVQ wurde gezeigt, dass dieses Verfahren eine bessere Auslastung der Verbindung und geringere Paketverluste als RED aufweist. Es ist jedoch zu beachten, dass diese Versuche in kontrollierten Umgebungen durchgeführt wurden mit dem Ziel zu zeigen, dass das jeweilige Verfahren das beste ist.

## VI. AUSBLICK UND ANDERE ANSÄTZE

Neben dem hier vorgestellten Ansatz des Active Queue Managements gibt es auch andere Verfahren, die zum Ziel haben, Überlastungen in Netzen zu vermeiden. In [14] werden hierzu zum Beispiel Zugangssteuerung und Routing unter Verkehrsberücksichtigung genannt. Bei der Zugangssteuerung werden nur dann neue, virtuelle Verbindungen aufgebaut, wenn das Netz diese auch verkraften kann. Beim Routing unter Verkehrsberücksichtigung werden Algorithmen angewandt, die im Netz nach möglichst wenig ausgelasteten Pfaden suchen. Diese Ansätze helfen somit auch, die Queues der Router zu entlasten.

Ein weiterer Ansatz, der den AQM Verfahren ähnlich ist, wird auch in [14] erwähnt: sogenannte Drosselpakete. Hier drosselt der Router den Verkehr nicht, indem er mit den Sendern durch das Markieren oder Fallenlassen von Paketen kommuniziert, sondern er gibt diese Nachricht über die anderen Router an den Sender zurück. Wenn der Router ausgelastet ist, schickt er ein Drosselpaket an den Router, von dem er das letzte Paket erhalten hat. Dieser reagiert sofort darauf, indem er weniger Pakete an diesen Router weiterleitet und seinerseits ein Drosselpaket an den Router vor ihm schickt. So kommt die Nachricht schneller beim Sender an und der Router wird schneller entlastet.

AQM Algorithmen sind ein aktuelles Thema der Forschung. Auch in naher Zukunft wird es wohl noch weitere neue Algorithmen geben, die bessere Eigenschaften als die bisherigen

aufweisen. Aktuell wichtiger für die Nutzer wäre jedoch, dass AQM Algorithmen auch wirklich auf allen Routern implementiert werden. Wie in dieser Arbeit gezeigt wäre das eine Verbesserung hin zur bestmöglichen Ausnutzung der heute angebotenen Ressourcen.

## LITERATUR

- [1] S. Floyd und K. Fall, "Router mechanisms to support end-to-end congestion control," Lawrence Berkeley National Laboratory, Berkeley CA, Tech. Rep., 1997.
- [2] K. Graffi, K. Pussep, N. Liebau, und R. Steinmetz, "Taxonomy of active queue management strategies in context of peer-to-peer scenarios," Technische Universität Darmstadt, Tech. Rep., 2007.
- [3] L. Le, J. Aikat, K. Jeffay, und F. Smith, "The effects of active queue management on web performance," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Ser. SIGCOMM '03. ACM, 2003.
- [4] K. Ramakrishnan, S. Floyd, und D. Black, "The addition of explicit congestion notification (ecn) to ip," United States, 2001.
- [5] J. Crowcroft und P. Oechsli, "Differentiated end-to-end internet services using a weighted proportional fair sharing tcp," *SIGCOMM Comput. Commun. Rev.*, vol. 28, Nr. 3, 1998.
- [6] R. Morris, "Tcp behavior with many flows," in *Proceedings of the 1997 International Conference on Network Protocols (ICNP '97)*, Ser. ICNP '97. IEEE Computer Society, 1997.
- [7] B. Suter, T. Lakshman, D. Stiliadis, und A. Choudhury, "Design considerations for supporting tcp with per-flow queueing," in *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, 1998.
- [8] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, und L. Zhang, "Recommendations on queue management and congestion avoidance in the internet," United States, 1998.
- [9] J. Nagle, "Congestion control in ip/tcp internetworks," *SIGCOMM Comput. Commun. Rev.*, vol. 14, Nr. 4, 1984.
- [10] W. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," United States, 1997.
- [11] R. Jain, "Congestion control in computer networks: issues and trends," *Network, IEEE*, vol. 4, Nr. 3, 1990.
- [12] S. Floyd, "Tcp and explicit congestion notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, Nr. 5, 1994.
- [13] R. Jain, "Congestion control and traffic management in atm networks: Recent advances and a survey," *Comput. Netw. ISDN Syst.*, vol. 28, Nr. 13, 1996.
- [14] A. Tannenbaum und D. Wetherall, *Computernetzwerke*. Pearson Deutschland GmbH, 2012.
- [15] T. Socolofsky und C. Kale, "Tcp/ip tutorial," RFC 1180 (Informational), Internet Engineering Task Force, 1991.
- [16] S. Oldak, W. Gong, C. Hollot, D. Towsley, V. Misra, und Y. Chait, "Active queue management for differentiated services," 2006.
- [17] G. Appenzeller, I. Keslassy, und N. McKeown, "Sizing router buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 34, Nr. 4, 2004.
- [18] B. Suter, T. Lakshman, D. Stiliadis, und A. Choudhury, "Efficient active queue management for internet routers," in *Proceedings of INTEROP, Engineering Conference*, 1998.
- [19] R. Pan, B. Prabhakar, und K. Psounis, "Active queue management toward fair bandwidth allocation," 2008.
- [20] A. Demers, S. Keshav, und S. Shenker, "Analysis and simulation of a fair queueing algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 19, Nr. 4, 1989.
- [21] C. Hollot, V. Misra, D. Towsley, und W.-B. Gong, "On designing improved controllers for aqm routers supporting tcp flows," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2001.
- [22] V. Firoiu und M. Borden, "A study of active queue management for congestion control," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2000.
- [23] S. Floyd, R. Gummadi, und S. Shenker, "Adaptive red: An algorithm for increasing the robustness of red's active queue management," AT&T Center for Internet Research at ICSI, Tech. Rep., 2001.
- [24] R. Pan, B. Prabhakar, und K. Psounis, "Choke - a stateless active queue management scheme for approximating fair bandwidth allocation," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, 2000.
- [25] S. Floyd und V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, Nr. 4, 1993.
- [26] W. Feng, K. Shin, D. Kandlur, und D. Saha, "Blue: A new class of active queue management algorithms," 1999.
- [27] S. Kunniyur und R. Srikant, "Analysis and design of an adaptive virtual queue (avq) algorithm for active queue management," *SIGCOMM Comput. Commun. Rev.*, vol. 31, Nr. 4, 2001.
- [28] S. McCanne, S. Floyd, und K. Fall, ns version 1 - lbnl network simulator. [Online]. Available: <http://ee.lbl.gov/ns/>
- [29] S. Floyd. (1997) Red: Discussions of setting parameters. [Online]. Available: <http://www.aciri.org/floyd/REDparameters.txt>
- [30] K. Ramakrishnan, A. L. Research, und S. Floyd, "A proposal to add explicit congestion notification (ecn) to ip," United States, 1999.