Staukontrolle durch Active Queue Management

Thomas Fischer, Dominik Billing

Masterseminar Kommunikationssysteme
Lehr- und Forschungseinheit für Kommunikationssysteme und Systemprogrammierung
Ludwig-Maximilians-Universität München

***Zusammenfassung*—Dieser Artikel beschreibt die Problematik im Internet, die durch den Einsatz von konventioneller Staukontrolle in Routern und der Struktur des Internets hervorgerufen wird. Diese liegt darin, dass Pakete nicht gleichmäßig fallen gelassen werden und es auf diese Art zu großem Overhead kommt, der durch Flaschenhälse im Internet noch verstärkt wird. Zusätzlich können besonders bandbreitenhungrige Datenströme 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 Staus zu verhindern. Zusätzlich werden alle Datenströme gleich behandelt. 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, BLUE und PI vorstellen und mit untereinander und mit ECN vergleichen.

I. Einführung und Motivation

Nach Floyd [1] sind Ende-zu-Ende (E2E) Staukontrollmechanismen von TCP mittlerweile ein kritischer Faktor in der Robustheit des Internets. Das Internet wächst unaufhaltsam weiter, es gibt keine eng verknüpfte Netzgemeinschaft mehr und nicht jeder Endknoten verwendet die E2E Staukontrolle für bestmöglichen Datenfluss. Entwickler kümmern sich nicht länger darum, E2E Staukontrolle in ihre Internet-Anwendungen zu integrieren. Die Konsequenz davon ist, dass das Netzwerk selbst seine Resourcennutzung kontrolliert.

Graffi [2] beschreibt, dass Bandbreite aktuell die knappste Ressource in Netzwerken ist. Die eingehenden und ausgehenden Bandbreiten normaler ADSL Verbindungen sind unterschiedlich. Typischerweise ist die ausgehende Bandbreite deutlich kleiner als die eingehende. Das bedeutet, dass nicht alle eingehenden Daten verarbeitet und weitergesandt werden können. Staus sind also Probleme, die hier auftreten können, wenn keine angemessenen Mechanismen angewendet werden. Das TCP-Protokoll sieht vor, Pakete nach dem ”First in First out”Prinzip fallenzulassen, wenn die Puffer voll laufen.

Die Problemstellung hier ist es also Mechanismen zu finden, Staus in E2E Verbindungen zu kontrollieren. Nach Le [3] soll die durchschnittliche Pufferausnutzung klein gehalten werden, damit E2E-Staukontrolle ermöglicht wird. Die Herausforderung besteht darin, dass an den Flaschenhälsen der potenzielle Stau erkannt wird und das komplette Netzwerk darauf reagiert, indem beispielsweise die Sendegeschwindigkeit des Ursprungs eines Datenstroms drastisch reduziert wird [2].

Das folgende Kapitel (Kapitel II) beschreibt die generellen Konzepte der Staukontrolle in Netzen und präsentiert Active Queue Management als Lösungsansatz für das Problem. In Kapitel III wird Active Queue Management definiert und einige Anwendungsfälle davon präsentiert. Die Active Queue Management Algorithmen RED, BLUE, ECN und PI werden in Kapitel IV vorgestellt und in Kapitel V miteinander verglichen. Ein Ausblick zu zukünftigen Entwicklungen und Forschungen sowie andere Ansätze zur Staukontrolle zusammen mit den zusammengefassten Ergebnissen erfolgt in Kapitel VI.

II. Staukontrolle in Netzen

Neben [1] wird auch von Morris [4] postuliert, dass die Effektivität von TCP mit zunehmender Anzahl an konkurrierender Datenströme nachlässt. Dieser Effekt beginnt sich zu zeigen, sobald mehr Datenströme als Pakete die vorhandene Bandbreite verwenden. 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 Haupzielen. 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. Zusätzlich soll die zur Verfügung stehende Bandbreite gleichmäßig zwischen konkurrierenden Datenströmen aufgeteilt werden [4]. Um diese gleichmäßige Verteilung zu erreichen, ist TCP standardmäßig mit dem „First Come First Serve“-Prinzip nicht geeignet. Deshalb müssen hierfür andere Methoden entwickelt werden, die den Puffer in Routern anders abarbeiten und dennoch einfach zu verwenden sind [5].

Die Internet Protokoll Architektur basiert auf einem verbindungslosen E2E Paketdienst, der das IP Protokoll benutzt. Die vielen Vorteile dieses verbindungslosen Designs, der Flexibilität und Robustheit wurden schon oft beschrieben. Allerdings kommen diese Vorteile zu einem Preis. Sorgfältiges Design ist benötigt, um einen guten Dienst zu leisten bei hoher Last. Fehlende Aufmerksamkeit bei den Dynamiken des Paketweiterleitens kann in ernsthafter Dienstdegradation enden oder „Internet Zusammenbruch“ [6]. Dieses Phänomen wurde während der ersten Wachstumsphase des Internets in den 1980er Jahren festgestellt und wird „congestion collapse“ [7] genannt. Bereits 1986 wurden von Jacobson entwickelte Stauverhinderungsmechanismen für Hosts entwickelt, die auch aktuell einen „congestion collapse“ verhindern [6].

Da das Internet seit dieser Zeit immer weiter wächst, wurde es offensichtlich, dass TCP Stauverhinderungsmechanismen [8], die absolut wichtig, nötig und mächtig sind, nicht unter allen Umständen ausreichend gute Dienste leisten. Das Hauptproblem liegt darin, dass von den Enden der Netzwerke nur bedingt Kontrolle ausgeübt werden kann. Deshalb müssen auch in Routern Mechanismen angewendet werden, die die Stauverhinderungsmechanismen der Endpunkte ergänzen. Hierbei muss man zwischen den beiden Klassen „Queue Management“ und „Scheduling“ von Router Algorithmen unterscheiden. 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 Bandbreite zwischen den Datenströmen zu verwalten [6].

Nach Jain [9] gibt es zwei Gründe, warum das Problem der Staukontrolle in Netzwerken sehr schwierig ist. Erstens gibt es Voraussetzungen für Staukontroll Schemas, die es schwierig machen eine zufriedenstellende Lösung zu finden. Zweitens gibt es unzählige Netzwerkregeln, die das Design eines Stauschemas beeinflussen. Das sind die Gründe, warum ein Schema, das für ein Netzwerk entwickelt wurde, in einem anderen Netzwerk nicht funktioniert. Grundbedingung für Schemas zur Staukontrolle sind: Das Schema muss einen kleinen Overhead haben, alle Datenströme gleichbehandeln, schnell auf andere Situationen reagieren können, in schlechten Umgebungen funktionsfähig sein und 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 TCP/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. Prinzipiell ist es in Netzwerken, in denen es die Hauptaufgabe von Routern ist, Pakete an den Output-Port weiterzuleiten, kein Problem Pakete einfach fallen zulassen. Dies ist allerdings im Internet heute nicht mehr der Fall, da E2E-Verbindungen über sehr viele Router gehen können. ECN ist ein zuverlässiger Mechanismus, beinhaltet allerdings keine Methoden zur Erkennung von Staus [10]. Da es nicht vorhersehbar ist, wie hoch der Datenverkehr zukünftig sein wird, besteht das wirkliche Problem also darin, Staus frühzeitig und zuverlässig zu erkennen [11].

Router sollten den Ursprüngen eines Datenstrom signalisieren, dass die Sendegeschwindigkeit reduziert werden muss, um Staus zu vermeiden. Zusätzlich kann so eine gleichmäßige Verteilung der Bandbreite erfolgen, wenn besonders gierige Datenströme vorhanden sind. „Active Queue Management“ (AQM) hat das Ziel Staus in Netzwerken rechtzeitig zu entdecken, bevor die Routerpuffer volllaufen. Es gibt im Prinzip zwei Möglichkeiten andere Router darüber zu informieren, dass ein Stau bevorsteht: Pakete können fallengelassen oder markiert werden. Die erste Strategie erfordert, dass die Endpunkte kooperieren und erzeugt einen Overhead, weil Pakete anschließend erneut gesendet werden müssen. Endpunkte müssen auf markierte Pakete reagieren, als wären sie fallengelassen worden, allerdings wird hier kein weiterer Overhead erzeugt, da die Pakete nicht erneut gesendet werden müssen. AQM-Algorithmen können zusätzlich die Bandbreite von besonders gierigen Datenströmen reduzieren, indem deren Pakete häufiger fallengelassen werden [2].

III. Definition und Anwendung von Active Queue Management

- Wirklich gute Quelle hierfür ist [6] Kapitel 2 ***wirklich sehr gute Quelle***
- Effizientes Active Queue Management in Internet Routern [12]
- Dimensionierung von Router Puffern [13]
- Stochastische Modellierung und die Theorie von Queues [14]
- Analyse und Simulation eines gleichbehandelnden Queue Algorithmus [15]

IV. Die gängigsten Active Queue Management Algorithmen

Seitdem die ersten Ideen für Active Queue Management vorgestellt wurden und mit der Einführung von ECN in IP/TCP wurden viele verschiedene AQM Algorithmen entwickelt. Es würde weit über den Rahmen dieser Arbeit hinausgehen, sie alle zu erwähnen, weshalb hier nur drei vorgestellt werden.

A. RED: Random Early Detection

Der erste Algorithmus, der präsentiert wird, ist Random Early Detection (RED). Er war einer der ersten AQM Algorithmen und viele andere Arbeiten entwickelten diesen weiter wie z.B. in [16] oder [17]. Andererseits werden Algorithmen, die anders ablaufen, oft mit RED verglichen.

Floyd und Van Jacobson haben RED 1993 vorgestellt [18]. In ihrer Arbeit wird die Funktionsweise des Algorithmus dargestellt. Um die Sender über die Verstopfung zu informieren kann RED entweder Pakete fallen lassen oder das ECN-Bit im Header setzen, je nach Router. Im folgenden werden wir nur die Möglichkeit des Markierens betrachten, was auf den eigentlichen Algorithmus keinerlei Auswirkungen hat. 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 Queuelänge Q_{min} und der maximalen Queuelänge 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 wie folgt aus der minimalen und maximalen Queuelänge Q_{min} und Q_{max}, der Durchschnittslänge Q_{avg} und dem Maximum für p_b, max_b, wie folgt: $p_b = max_p \frac{Q_{avg}-Q_{min}}{Q_{max}-Q_{min}}$. 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: $p_a = \frac{p_b}{1-zp_b}$. Der Zähler wird für jedes einkommende Paket inkremetiert. Ein Paket wird mit der Wahrscheinlichkeit p_a markiert. Ist dies der Fall, so wird der z auf 0 gesetzt. Mithilfe des Zählers steigt die Wahrscheinlichkeit somit für jedes weitere Paket an.

Die Markierungswahrcheinlichkeit für ein Paket einer Verbindung verhält sich also in etwa proportional zum Anteil der Bandbreite, den diese Verbindung belegt. Auf diese Weise versucht der Algorithmus die Resourcenzuteilung fair zu machen. Neben der Queuelänge in Paketen könnte RED auch die Bytelänge und somit die Anzahl an Bytes eines Pakets zur Bewertung heranziehen. Diese Information wird dann in den Markierungswahrscheinlichkeit p_b mit einbezogen. Nach der Berechnung ändert sich der Wert dann in $p_b = p_b \frac{Paketbytes}{maximalePaketbytes}$. Dadurch werden große Pakete mit einer höheren Warscheinlichkeit markiert als kleine Pakete. An der Beschreibung des Algorithmus wird ersichtlich, dass RED viele Parameter benötigt, die vorab festgelegt werden müssen. Damit RED gut läuft, müssen für jeden Router die richtigen Werte gefunden werden, was ein nicht zu vernachlässigendes Problem des Algorithmus darstellt. Mehr dazu in Abschnitt V.

B. BLUE

Der BLUE Algorithmus wurde 1999 von Feng et.al. an der University of Michigan in Zusammenarbeit mit IBM vorgestellt [19]. BLUE wurde entwickelt, um einige Schwachstellen von RED zu verbessern, ist aber ein völlig neuer Ansatz. Wie bereits erläutert verlässt sich RED auf die Queuelänge, um Verstopfungen anzuzeigen, 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 Markierungswahrscheinlichkeiten zu berechnen. Genause wie RED kann BLUE dann entweder Pakete fallen lassen oder mittels ECN markieren.

Der BLUE Algorithmus kennt nur eine Markierungswahrscheinlichkeit p_m; jedes einzureihende 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 der Algorithmus folgendermaßen ab:

$if(Paketverlust \wedge (now - last\_update) > freeze\_time)\ then$
  $p_m = p_m + d_1$
  $last\_update = now$
$if(Verbindung\ \ frei \wedge (now - last\_update > freeze\_time))\ then$
  $p_m = p_m - d_2$
  $last\_update = now$

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 Verstopfungen sehr viel schneller reagieren kann. Die Autoren geben weiterhin an, dass in ihren Experimenten die minimale Zeitspanne zwischen Änderungen an p_m, 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.

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 [20]. Wie bereits der Name andeutet basiert der Algorithmus auf einer virtuellen 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 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 $\dot{C}_v = \alpha(\gamma C - \lambda)$ wobei α ein Glättungsparameter, γ die angestrebte Auslastung der Verbindung und λ die Ankunftsrate der Verbindung ist. Das Markieren passiert auf diese Weise aggressiver, also häufiger, wenn die Verbindung ihre gewünschte Bandbreite 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 als Algorithmus konkret implementiert werden kann:

Für jedes ankommende Paket DO
$VQ = max(VQ - C_v(t-s), 0)$
$if(VQ + b > B)\ then$
  Paket markieren
else
  $VQ = VQ + b$
$C_v = max(min(C_v + \alpha \cdot \gamma \cdot C(t-s), C) - \alpha \cdot b, 0)$
$s = t$

wobei *B* die Puffergröße in Bytes, s die Ankunftszeit des letzten Pakets, t die aktuelle Zeit, b die Paktegröße des

aktuellen Pakets in Bytes und *VQ* die Anzahl an Bytes, die aktuell in der virtuellen Queue sind, ist.

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 α und die gewünschte Ausnutzung γ vorab angegeben werden. Diese dienen der Stabilität des Verfahrens. Weiter geben die Autoren an, dass γ es den ISP's erlaubt einen Ausgleich zwischen hoher Bandbreitenausnutzung und kleinen Puffern vorzunehmen. Eine Regel, wie diese Parameter gesetzt werden sollten, befindet sich in der Originalarbeit in [20].

V. Vergleich der vorgestellten Algorithmen

- Vergleich RED, ARED, PI [3]
- Vergleich RED, PI [20]
- Vergleich RED, BLUE, ARED, ECN, PI [2]

VI. Ausblick und andere Ansätze

- Ein wirklich optimaler Algorithmus muss noch gefunden werden [2]
- Statt Staukontrolle andere Wege suchen (CHOKe) [17]

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] 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.

[5] 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, Mär. 1998, pp. 299–306vol.1.

[6] 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.

[7] J. Nagle, "Congestion control in ip/tcp internetworks," *SIGCOMM Comput. Commun. Rev.*, vol. 14, Nr. 4, pp. 11–17, Okt. 1984. [Online]. Available: http://doi.acm.org/10.1145/1024908.1024910

[8] W. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," United States, 1997.

[9] R. Jain, "Congestion control in computer networks: issues and trends," *Network, IEEE*, vol. 4, Nr. 3, pp. 24–30, Mai 1990.

[10] S. Floyd, "Tcp and explicit congestion notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, Nr. 5, pp. 8–23, Okt. 1994.

[11] R. Jain, "Congestion control and traffic management in atm networks: Recent advances and a survey," *Comput. Netw. ISDN Syst.*, vol. 28, Nr. 13, pp. 1723–1738, Okt. 1996. [Online]. Available: http://dx.doi.org/10.1016/0169-7552(96)00012-8

[12] B. Suter, T. Lakshman, D. Stiliadis, und A. Choudhury, "Efficient active queue management for internet routers," in *Proceedings of INTEROP, Engineering Conference*, 1998.

[13] G. Appenzeller, I. Keslassy, und N. McKeown, "Sizing router buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 34, Nr. 4, pp. 281–292, Aug. 2004.

[14] R. Wolff, *Stochastic modeling and the theory of queues*. Prentice Hall, 1998.

[15] A. Demers, S. Keshav, und S. Shenker, "Analysis and simulation of a fair queueing algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 19, Nr. 4, pp. 1–12, Aug. 1989.

[16] 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.

[17] 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, pp. 942–951vol.2.

[18] S. Floyd und V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, Nr. 4, pp. 397–413, Aug. 1993.

[19] W. Feng, K. Shin, D. Kandlur, und D. Saha, "Blue: A new class of active queue management algorithms," 1999. [Online]. Available: http://www.eecs.umich.edu/techreports/cse/99/CSE-TR-387-99.pdf

[20] 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, pp. 123–134, Aug. 2001.

[21] S. Athuraliya, S. Low, V. Li, und Q. Yin, "Rem: active queue management," *Network, IEEE*, vol. 15, Nr. 3, pp. 48–53, Mai 2001.

[22] W. Feng, K. Shin, D. Kandlur, und D. Saha, "The blue active queue management algorithms," *IEEE/ACM Trans. Netw.*, vol. 10, Nr. 4, pp. 513–528, Aug. 2002.

[23] 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, Mär. 2000, pp. 1435–1444vol.3.

[24] 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, pp. 1726–1734vol.3.

[25] K. Ramakrishnan, S. Floyd, und D. Black, "The addition of explicit congestion notification (ecn) to ip," United States, 2001.