ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Netzwerkstaukontrolle und die Arbeitsweise verschiedener Algorithmen in TCP

von

Michał Roziel

Matrikelnummer : 5012845

Ein wissenschaftlicher Bericht im Rahmen der Vorlesung

„Wissenschaftliches Arbeiten“

an der htw saar im Studiengang Informatik

Saarbrücken, den 28. August 2025

Abstract

Dieser Bericht zielt darauf ab, aktuell benutzte Algorithmen der Netzwerkstaukontrolle in Computernetzwerken zu vergleichen.

Die Algorithmen der Staukontrolle, welche in diesen Vergleich einfließen sind : TCP BBR, TCP NewReno, TCP Cubic, und TCP Vegas.

Mittels des Open-Source Netzwerksimulators *NS-3* wird ein virtuelles Netzwerk mit einer 10Mbit/s Engstelle zwischen zwei Endpunkten aufgestellt. In diesem Netzwerk wird zunächst ein künstlicher Datenverkehr erzeugt. Der genannte Datenverkehr wird anschließend aufgezeichnet und das dabei entstehende *Congestion Window*(CW) dient als Basis für den Vergleich. Die unterschiedlichen CW's werden schließlich analysiert und in Bezug auf das gegebene Szenario evaluiert. Dieser Bericht bietet dem Leser einen Überblick über einige CC-Algorithmen und ein grundlegendes Verständnis darüber, wie Computernetzwerke auf Datenstau und Paketverlust reagieren.

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Insbesondere habe ich alle KI-basierten Werkzeuge angegeben, die ich bei der Erstellung, Übersetzung oder Überarbeitung des Textes verwendet habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, den 28. August 2025

Unterschrift Michał Roziel

Inhaltsverzeichnis

1. Einleitung

Network congestion control (CC) ist ein essentieller Bestandteil der meisten modernen Computernetzwerke. Wenn eine Netzwerkschnittstelle zu einem Zeitpunkt versucht eine zu große Menge an Datenpaketen aufzunehmen, kommt es zu Stau von Datenverkehr und zu einem potentiellen Verlust von Datenpaketen. Aufgrund diesem Vorkommen werden Algorithmen innerhalb von Netzwerkprotokollen verwendet, diese erkennen den Anstau von Datenverkehr im Netzwerk, und helfen den Fluss von Datenpaketen zu steuern. Neben dem effizienten Durchfluss von Informationen ist zeitgleich auch die Fairness, was die Verteilung von Ressourcen einer Netzwerkschnittstelle an ihre Hosts angeht, wichtig. Auch dies wird von CC-Algorithmen gewährleistet. In dem heutigen Stand von Rechnernetzen werden verschiedene Protokolle zur Netzwerkstaukontrolle verwendet, Ich werde mich im Rahmen dieses Berichts allerdings auf das Transmission Control Protocoll beschränken, da dies das am meisten verbreitete ist.

Mit stets weiterentwickelten Rechnernetzen, und einem jährlich zunehmenden Datenverkehr, wie auch in der Deutschen Internetschnittstelle *DE-CIX*[1] in Frankfurt gewinnen diese Algorithmen an Bedeutung.

1.1. Grundlagen und Begriffe

Um die Funktionsweise und den Unterschied zwischen den hier behandelten CC-Algorithmen verstehen zu können, muss der Leser erst ein Grundverständnis über die Kommunukation innerhalb von Computernetzwerken haben. Grundsätzlich wird heutzutage eins von zwei am verbreitesten Protokollen verwendet, entweder TCP oder UDP.

1.1.1. Transmission Control Protocoll

Das Transmission Control Protocoll (TCP) ist ein weit verbreitetes Netzwerkprotokoll. TCP lässt sich in zu der Schicht 4 (Transportschicht) in dem OSI-Modell einordnen, hierbei liegt es zwichen der Vermittlungs- und Kommunikationsschicht. TCP wurde 1981 unter RFC 793 erstmals standarisiert. [2]
Die Aufgabes des TCP Protokolls ist es, zwischen zwei Hosts eine Verbindung aufzubauen, welche anschließend dazu genutzt wird, Nachrichten in zu verschicken und zu empfangen.

> "transport-layer protocol [...] from an application's perspective, it is as if the hosts running the processes were directly connected;" [3, 241].

Dies Bedeutet, dass TCP ebenfalls eine gewisses Abstraktionsniveau des Nachrichtenaustausches abnimmt.

Ein klares Unterscheidungsmerkmal des TCP von dem ebenfalls bekannten User Datagramm Protocoll (UDP) ist, dass TCP verbindungsorientiert arbeitet, während UDP als verbindungslos gilt. Zudem werden bei TCP Daten in Form von Packets

verschickt, während man bei UDP von Datagrammen spricht.
Bei TCP wird zu Beginn ein *Three-Way-Handshake* wie folgt durchgeführt :

Der Sender schickt zunächst eine Nachricht mit einem Verbindungswunsch an den Empfänger und setzt das Flag *Synchronize,* ***SYN***.
Der Empfänger antwortet mit einer Bestätigung die erste Nachrichte erhalten zu haben, und schickt ebenfalls einen Verbindungswunsch. Es werden die Flags ***SYN-ACK*** (Synchronise-Acknowledge) gesetzt.

Als letztes schickt der Sender eine Bestätigung, dass die Nachricht des Empfängers angekommen ist, dies geschieht wieder mit dem Flag ***ACK***. Nun ist die Verbindung bereit, Daten in beide Richtungen zu übertragen. Nach jedem gesendeten Datenpaket folgt eine ***ACK*** Bestätigung.
Die Abbildung 2 im Anhang visualisiert den Three-Way—Handshake.

1.2. Algorithmen der Staukontrolle

Algorithmen der Staukontrolle kommen zum Einsatz, wenn in einem Netzwerk ein potentieller Datenstau erkannt wird. Grundsätzlich starten die Algorithmen in dem Arbeitsmodus *slow start*. Falls Datenstau erkannt wird, treten diese Algorithmen in die Phase *congestion avoidance*.

Hiermit wird mit verschiedenen Vorgehensweisen gegen die Netzwerküberlastung gesteuert.
Es existieren unterschiedliche Typen von CC-Algorithmen, die jeweils ihre eigenen Vor- und Nachteile haben. In diesem Bericht wird jeweils ein Algorithmus pro Typ behandelt.

1.2.1. TCP BBR

Der *Bottleneck Bandwith and Round-Trip-Time* Algorithmus wurde erstmals in dem von der *Association for Computing Machinery* (ACM) veröffentlichten Magazin ACM vorgestellt.[4].

Seit 2016 wurden 3 Versionen entwickelt. Diese werden jeweils mit $BBRv\_x$ gekennzeichnet, wobei x die entsprechende Version beschreibt.
BBR ist ein modellbasiertes Staukontrollverfahren, dessen Arbeitsweise in 4 Phasen unterteilt werden kann :

1. $STARTUP$(Startphase)
2. $DRAIN$(Abbau des Staus)
3. $PROBE\_BW$(Proben der Bandweite)
4. $PROBE\_RTT$(Messen der Round-Trip-Time)

Während andere CC-Algorithmen wie etwa Reno oder CUBIC auf Paketverlust reagieren, misst BBR die vorliegende Engpassbreite sowie RTT, und bestimmt daraus die Sende-Rate, welche anschließend in dem Netzwerk eingesetzt wird. Da BBR nicht

direkt auf Paketverlust reagiert, kommt es mit gut mit zufälligen Paketverlust zurecht, hat aber zeitgleich Schwierigkeiten mit der Fairness was die Verteilung von Ressourcen angeht :

> "Unfairness happens [...], which relates to different settings of round-trip times and buffer sizes. Essentially, the reason for the unstable or unequal unfairness is the absence of responding mechanisms for bandwidth convergence."[5]

Die Fairness kann hierbei mit dem zwischen 0 und 1 liegenden Jain's Fairness Index gemessen werden.[6]

1.2.2. TCP Reno

1.2.3. TCP Cubic

Der Algorithmus TCP Cubic ist ein Loss-Based CC-Algorithmus, das bedeutet der Arbeitsmodus *congestion avoidance* tritt ein, falls ein Paketverlust erkannt wird.

Die Wachstumsfunktion, welche anschließend die neue Größe des *Congestion Window* beschreibt, wird definiert als : $W(t) = C(t-K)^3 + W\max$. Hierbei ist W die größe des Congestion Windows, C eine Cubic-Konstante, t die Zeit seit dem letzten Paketverlust, K die Zeit, welche *TCP Cubic* benötigt um wieder $Wmax$ zu erreichen.

Mit der Sensitivität auf Paketverlust β kann K wie folgt berechnet werden :

$$K = \sqrt[3]{\frac{W_{\max} \cdot \beta}{C}}$$

1.2.4. TCP Vegas

1.2.5. Merkmale der CC Algorithmen auf einen Blick

	Algorithmus	**Typ**	**Signal der Congestion avoidance**
BBR	a	b	c
NewReno	d	e	f
Cubic	g	h	i
Vegas	n	f	f

1.3. Definitionen

1.3.1. Queuing Delay

Queuing Delay beschreibt das Warten einer *Queue* auf das Versenden von Packets durch das gegebene Netzwerk.

2. Versuchsaufbau

Mit einem Test innerhalb des Command Line Simulators *NS-3.45* lässt sich ein beliebieger CC-Algorithmus ausführen.

Netzwerktopologie

Zunächst definieren wir die Netzwerktopologie auf welcher die CC-Algorithmen laufen werden.[7]

```
            1000 Mbps           10 Mbps           1000 Mbps
Sender -------------- R1 -------------- R2 -------------- Receiver
            5 ms                10 ms              5 ms
```

So ist es möglich, einen Netzwerkstau zu simulieren, dieser bleibt für jeden Algorithmus gleich.

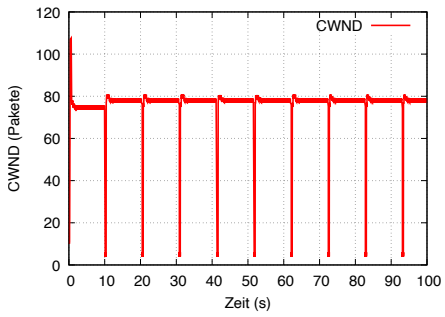
Durch das Verändern der folgenden *C++* Codezeile der Datei *tcp-bbr-example.cc* mit dem Namen des gewünschten Algorithmus lässt sich der Test unter gleichen Netzwerkbedingungen für die anderen Tesstfälle ausführen :

```
[...]
    std::string tcpTypeId = "ALGO_NAME";
[...]
```

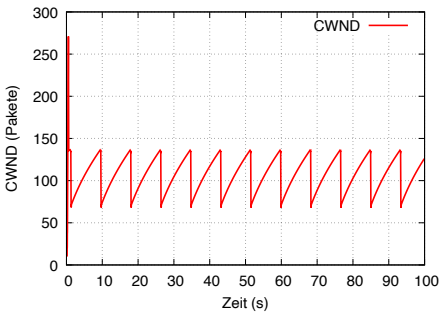
Mittels der Navigation in das Directory *ns-3.45* per Kommandozeile und dem Befehl

```
./ns3 run examples/tcp/tcp-bbr-example.cc
```

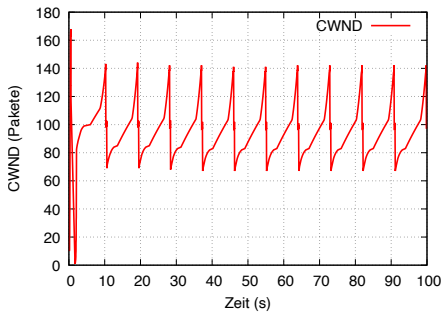
lässt sich der jeweilige Testlauf ausführen. Die aufgezeichneten Informationen können mit dem im Anhang in .2 hinterlegten Skript visualisiert werden.



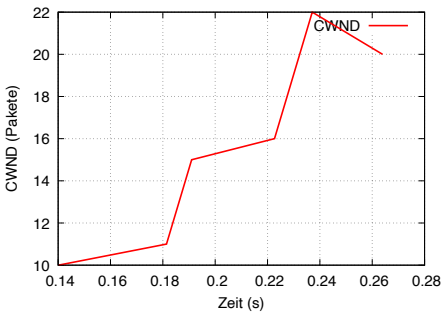
(a) BBR



(b) NewReno



(c) Cubic



(d) Vegas

Abbildung 1: Vergleich der aufgezeichneten Congestion Windows.

2.1. Analyse der Ergebnisse

Wie an den *Congestion Window* Graphen erkennbar, besitzt jeder der vorgestellten Algorithmen eine eigene Herangehensweise was das Regulieren von Datenstau angeht.

3. Ergebnisse und Diskussion

4. Fazit und Schlussfolgerungen

4.1. Offene Fragen

4.2. Diskussion

Literaturverzeichnis

[1] DE-CIX, "Traffic statistics frankfurt," 2025. Letzter Zugriff: 28. August 2025.

[2] J. Postel, "transmission control protocol," 1981.

[3] K. W. James F.Kurose, *Computer Networking: A Top-Down Approach*. London: Pearson, 8th ed., 2022.

[4] N. Cardwell, Y. Cheng, C. S. Gunn, S. Hassas Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, 2016.

[5] L. Tang, "Bbr fairness evaluation using ns-3," 2024.

[6] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *CoRR*, vol. cs.NI/9809099, 1998.

[7] M. P. T. Aarti Nandagiri, Vivek Jain, "Tcp bbr example," 2020. Copyright (c) 2018-20 NITK Surathkal.

Anhang

.1. Zusätzliche Abbildungen

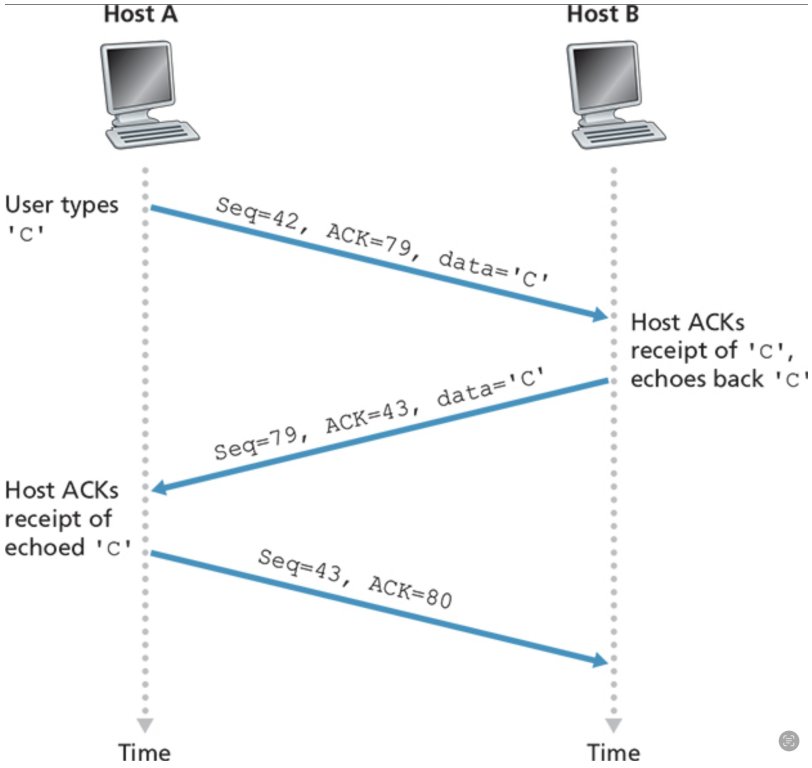


Abbildung 2: Detaillierte Darstellung des Three-Way Handshake.

.2. GnuPlot Skript zur Erstellung der CWND Graphen

```
set terminal pdfcairo size 10cm,7cm enhanced font "Helvetica,14"
set output "cwnd.pdf"

set xlabel "Zeit (s)"
set ylabel "CWND (Pakete)"
set grid
set key top right

plot "cwnd.dat" using 1:2 with lines lw 2 lc rgb "red" title "CWND"
```

A. Datenmaterial

B. Web-Standards

O