Vorlesung Kommunikationssysteme
Wintersemester 2024/25

Ausblick: TCP für Hochgeschwindigkeitsnetze

Christoph Lindemann

Zeitplan

Nr.	Datum	Thema
01	18.10.24	Organisation und Internet Trends
02	25.10.24	Programmierung mobiler Anwendungen mit Android
	01.11.24	Keine Vorlesung
03	08.11.24	Protokolldesign und das Internet
04	15.11.24	Anwendungen und Netzwerkprogrammierung
05	22.11.24	LAN und Medienzugriff
06	29.11.24	Ethernet und drahtlose Netze
07	06.12.24	LAN Komponenten und WAN Technologien
08	13.12.24	Internetworking und Adressierung mit IP
09	20.12.24	IP Datagramme
10	10.01.25	Zusätzliche Protokolle und Technologien
11	17.01.25	User Datagram Protocol und Transmission Control Protocol
12	24.01.25	TCP Überlastkontrolle / Internet Routing und Routingprotokolle
13	31.01.25	Ausblick: TCP für Hochgeschwindigkeitsnetze
14	07.02.25	Review der Vorlesung

Überblick

Ziele:

grundlegendes Verständnis der Überlastkontroll-mechanismen, die heute im Internet eingesetzt werden

Interesse an den weiterführenden Lehrveranstaltungen des Lehrstuhls wecken

Inhalte:

- Bezug zwischen TCP Überlastkontrolle und aktiver Warteschlangen-verwaltung
- Umfassende Behandlung von TCP Überlastkontrolle
- TCP CUBIC und BBR

Lehrangebot am Lehrstuhl RVS (1)

Sem	SS/ WS		LP(h)
Bachelorstudium			
3.	WS	Pflichtmodul: Kommunikationssysteme	5(150)
4.	SS	Kernmodul: Rechnernetze	5(150)
5.	WS	Seminarmodul: Rechnernetze und Internetanwendungen I	5(150)
6.	SS	Bachelorarbeit	

Grundlagen der Überlastkontrolle

Überlast:

- Umgangssprachlich: "Zu viele Quellen senden zu viele Daten zu schnell, um vom Netz transportiert zu werden"
- Ausprägungen:
  - Paketverlust (Pufferüberläufe an den Routern)
  - Große Verzögerungen (Dauer der Pufferung)
- Sehr wichtiges Problem!

Ansätze zur Überlastkontrolle

Zwei mögliche Herangehensweisen:

Ende-zu-Ende Überlastkontrolle:

- Keine direkte Rückmeldung vom Netz
- Die momentane Auslastung wird aus den vom Sender und Empfänger beobachtbaren Parametern ermittelt: Paketverlust, Verzögerungszeit
- Ansatz bei TCP

Netzgestützte Überlastkontrolle:

- Router liefern Rückmeldung an Sender und Empfänger:
  - Bit zeigt Überlast an (SNA, DECbit, TCP/IP ECN, ATM)
  - Bestimmung einer Rate, mit der Sender Pakete aussendet

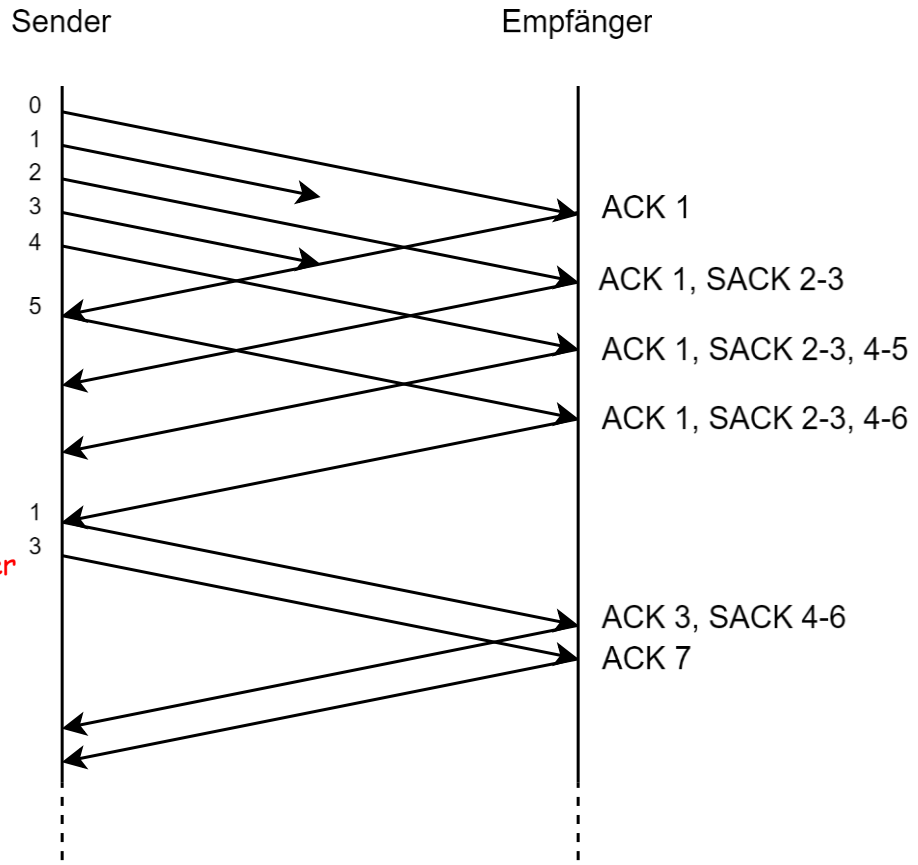
Explicit Congestion Notification

- Router können dem Sender Überlast signalisieren ohne Pakete zu verwerfen
- Existierende Pakete werden mit gesetzten Bits im Header markiert
  - ECN-Capable Transport (ECP)
  - Congestion Experienced (CE)
  - ECN-Echo (ECE)
- Sender setzt ECT-Bit in ECN-fähigen Paketen
- Router/Switch der Überlast erkennt markiert Pakete mit CE-Bit, anstatt sie zu verwerfen
- Empfänger meldet Überlast an Sender indem er ECE-Bit im TCP-Header des ACK setzt

Selective Acknowledgement

- Empfänger bestätigt selektiv korrekt empfangene Pakete
- Sender kann nur verlorene Pakete erneut senden anstatt alle unbestätigten Pakete
- SACK übermittelt Informationen im Option Part des TCP-Headers
- SACK 4-6
  - Paket 4 erhalten
  - Paket 6 erwartet

Viertes ACK 1!
Neuübertragung aller verlorenen Pakete.



Überblick

7.1 Einführung

7.2 Motivation

7.3 Zusammenspiel zwischen Überlastkontrolle und adaptiver Warteschlangen-verwaltung

7.4 Überlastkontrolle basierend auf Paketverlust

- CUBIC

7.5 Überlastkontrolle basierendauf Verzögerungszeit

- BBR

7.6 Maschinelle Lernverfahren für TCP

7.7 Reinforcement Learning for Dynamic Initial Window

7.8 Reinforcement Learning for Congestion Control

- TCP RL
- Pareto: Fair Congestion Control

Motivation

- Netzwerkeigenschaften haben sich drastisch verändert seit die Internetprotokolle entworfen wurden
- Gründe für die Veränderungen sind mehr Geräte, günstigere/leistungsfähiger Hardware und neue Forschungsergebnisse

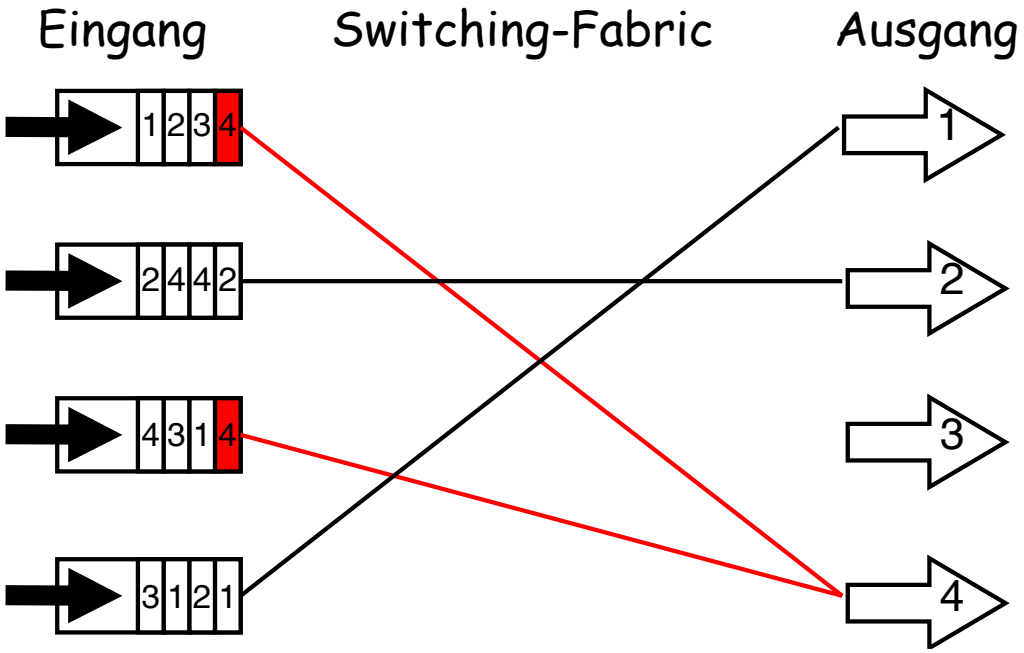
- Internet 1990
  - Etwa 300.000 Geräte
  - effektive Bandbreite < 30 Kbit/s
- Internet heute
  - > 1 Milliarde Geräte
  - effektive Bandbreite > 100 Mbit/s

Initiale Fenstergröße zu klein

- Recap SlowStart:
  - Congestion Window (cwnd) startet mit 1 (2 oder 10) MSS und wird exponentiell gesteigert, bis erster Verlust auftritt

- TCP-Verbindungen in Hochgeschwindigkeitsnetzen verbringen mehr Zeit in der Slow Start Phase als in der Congestion Avoidance Phase
- Bandbreite wird nicht effektiv ausgenutzt
- Wählt man die initiale Fenstergröße anders, sind bessere Ergebnisse zu erwarten

Head of Line Blocking

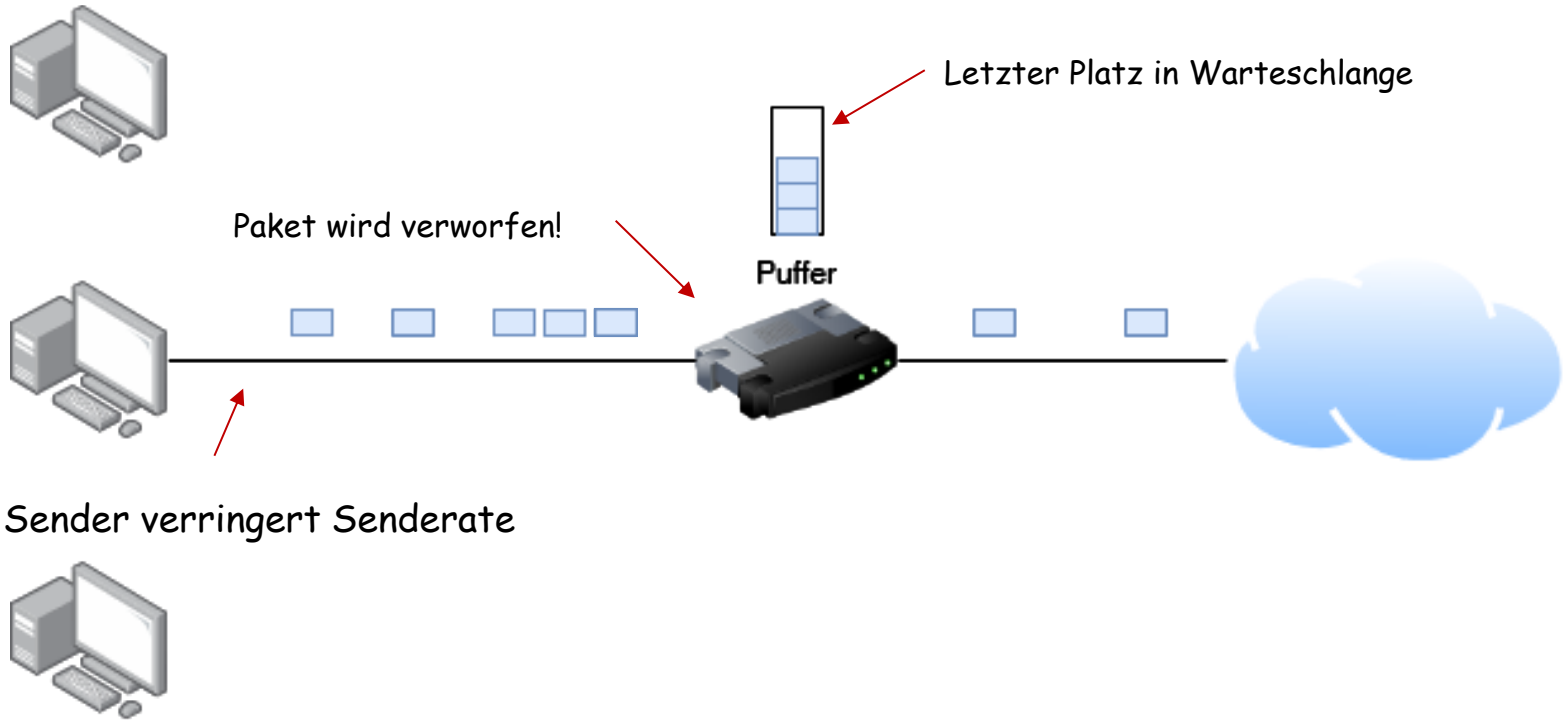


- Ein Paket in einer Warteschlange hindert nachfolgende Pakete, weitergeleitet zu werden, z.B. in Routern oder Switches

Aufblähen des Puffers (1)

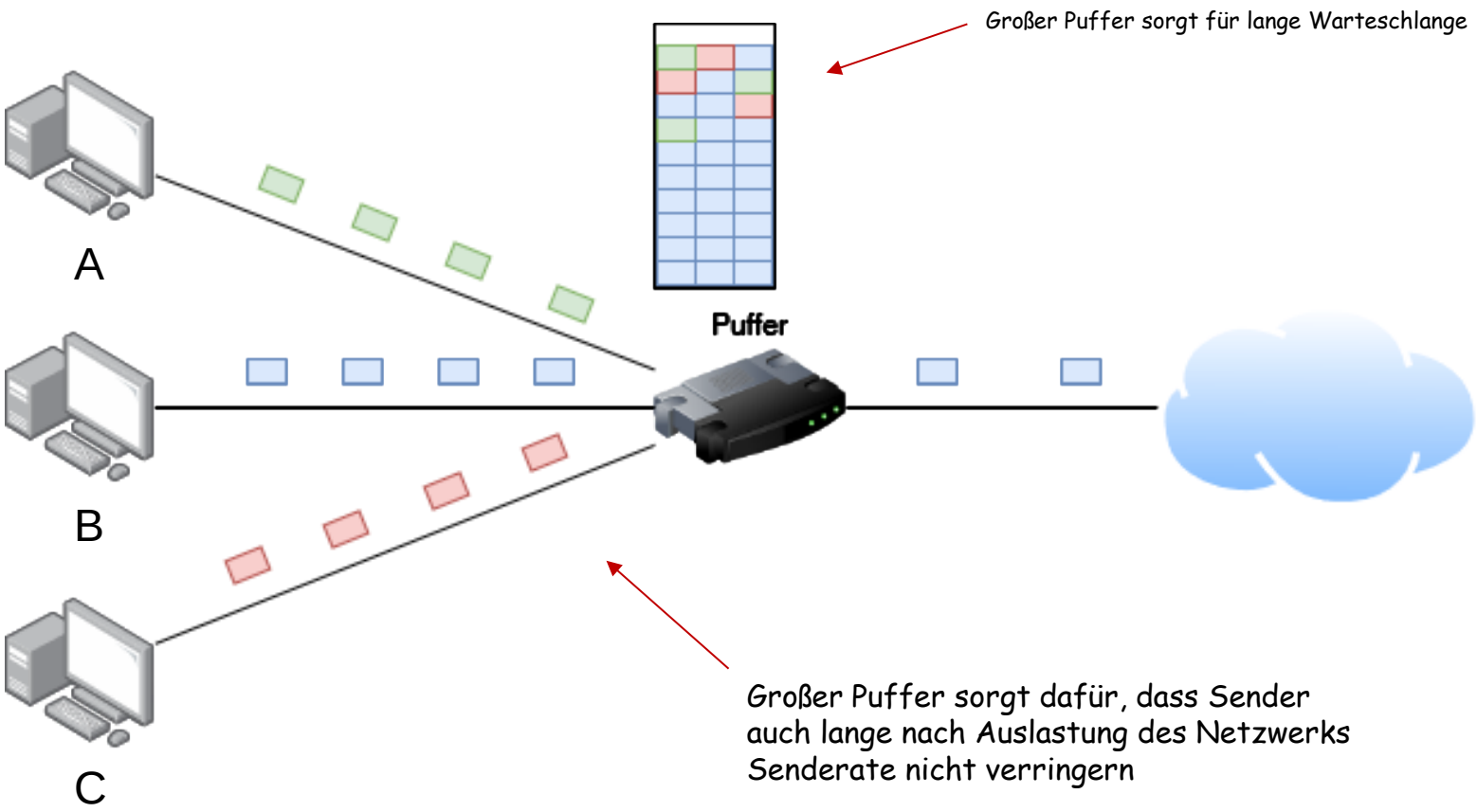
- Günstige Hardware sorgt für größere Pufferspeicher in Routern als zu den Anfängen des Internets
- Bei beginnender Überlast wird zuerst der Pufferspeicher gefüllt und Paketverluste treten verzögert auf
- Pakete verbringen daher viel Zeit in der Warteschlange (ie dem Pufferspeicher) bevor sie weitergeleitet werden → hohe Round Trip Times
- Besonders betroffen sind neue TCP Flüsse, welche über eine bereits überlastete Ende-zu-Ende Verbindung senden

Aufblähen des Puffers (2)



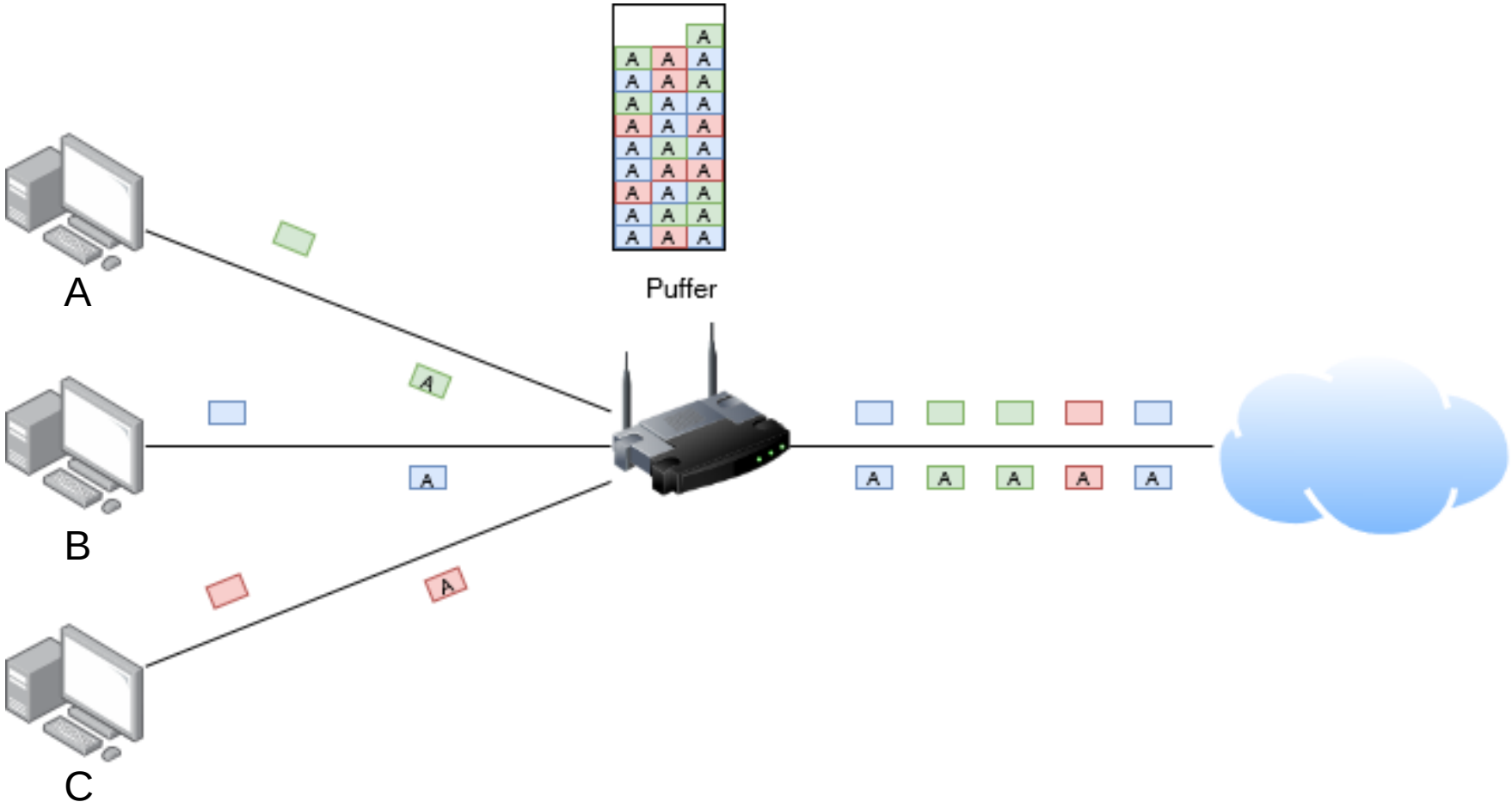
- TCP-Verbindung mit kleinem Puffer und Flaschenhals am Uplink des Routers
- Puffergröße begrenzt Anzahl an Paketen in der Warteschlange

Aufblähen des Puffers (3)



- TCP-Verbindung mit großem Puffer und Flaschenhals am uplink des Routers
- Pakete verbringen sehr lange in Warteschlange

Aufblähen des Puffers (4)



- TCP-Verbindung mit großem Puffer und Flaschenhals an WiFi Verbindung
- Acknowledgments verbringen sehr lange in Warteschlange

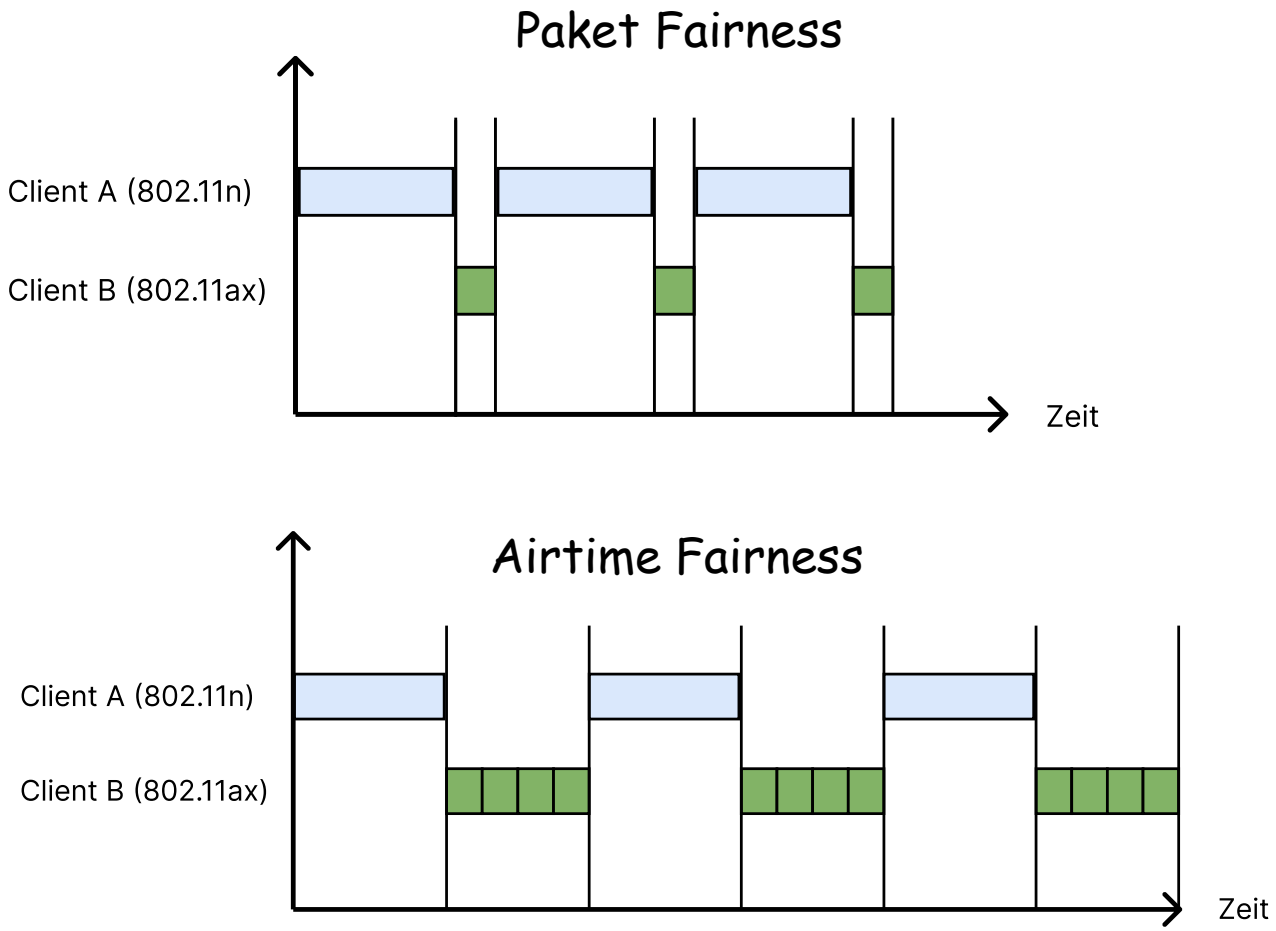
Airtime Fairness

- In drahtlosen Netzwerken können langsame Geräte mehr Zeit für das Senden von Daten benötigen
- Schnellere Geräte müssen auf langsame Geräte warten und erzielen gleichen Durchsatz wie langsame Geräte

- **Airtime Fairness** → alle Geräte sollten gleichberechtigt Sendezeit (Airtime) für Datenübertragungen erhalten

- Moderne Access Points (APs) verteilen Airtime gleichmäßig an alle verbundenen Geräte durch bevorzugte Behandlung schnellerer Geräte

Airtime Fairness



Überblick

7.1 Einführung

7.2 Motivation

7.3 Zusammenspiel zwischen Überlastkontrolle und adaptiver Warteschlangen-verwaltung

7.4 Überlastkontrolle basierend auf Paketverlust

- CUBIC

7.5 Überlastkontrolle basierend auf Verzögerungszeit

- BBR

7.6 Maschinelle Lernverfahren für TCP

7.7 Reinforcement Learning for Dynamic Initial Window

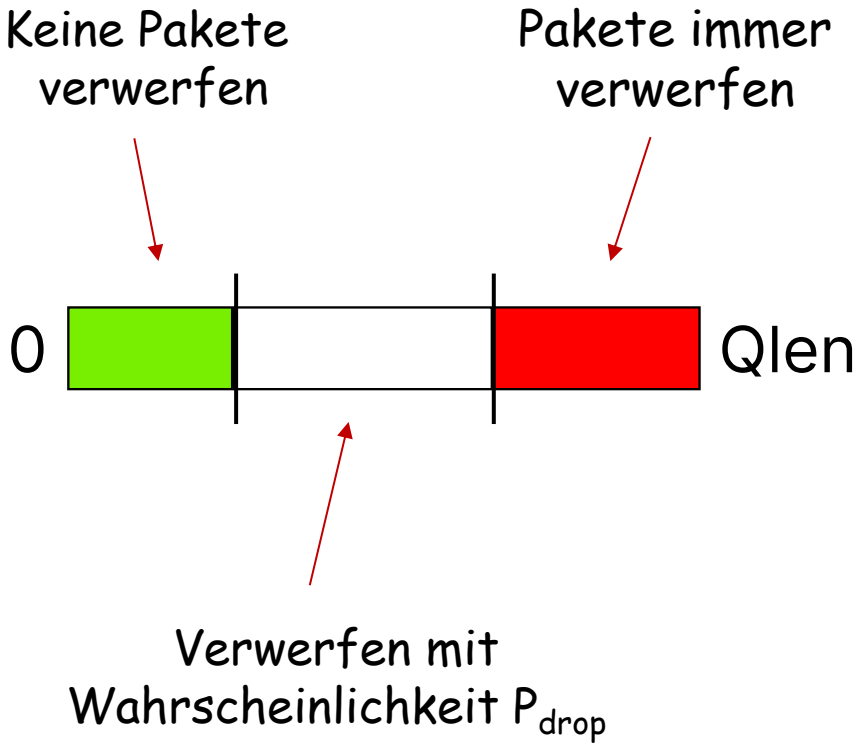
7.8 Reinforcement Learning for Congestion Control

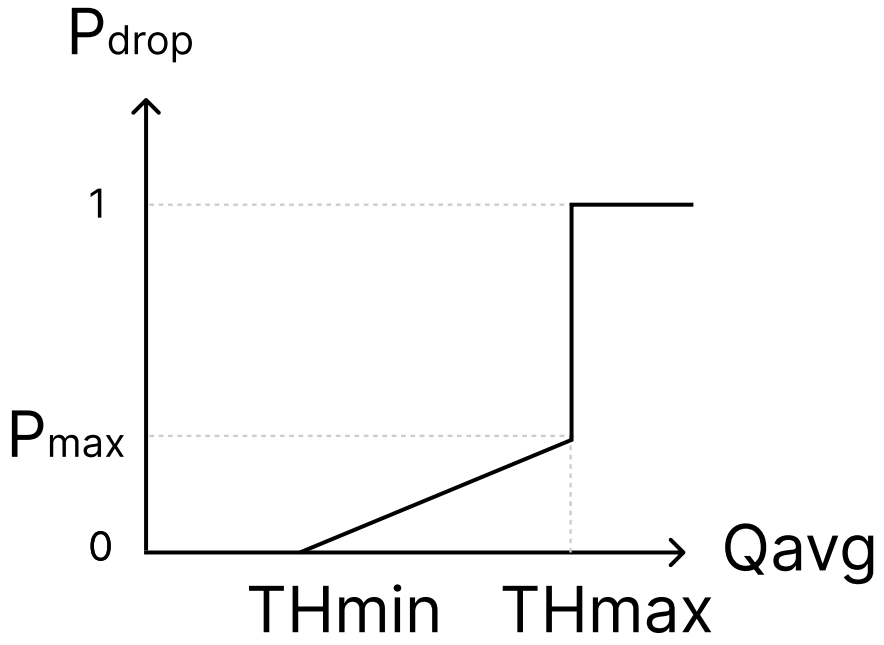
- TCP RL
- Pareto: Fair Congestion Control

Random Early Detection

- Adaptive Queue Management (AQM)
- RED ist ein Pufferverwaltungsalgorithmus, der in Routern verwendet wird um Netzüberlast zu verhindern
- Frühzeitiges Verwerfen von Paketen, bevor der Puffer vollständig ausgelastet ist
- Minimaler (THmin) und maximaler (THmax) Schwellwert
- Pakete bei Füllständen unterhalb THmin immer annehmen und oberhalb THmax immer verwerfen
- Zwischen Schwellwerten steigt die Verwerfungswahrscheinlichkeit (P_{drop}) linear an

Random Early Detection

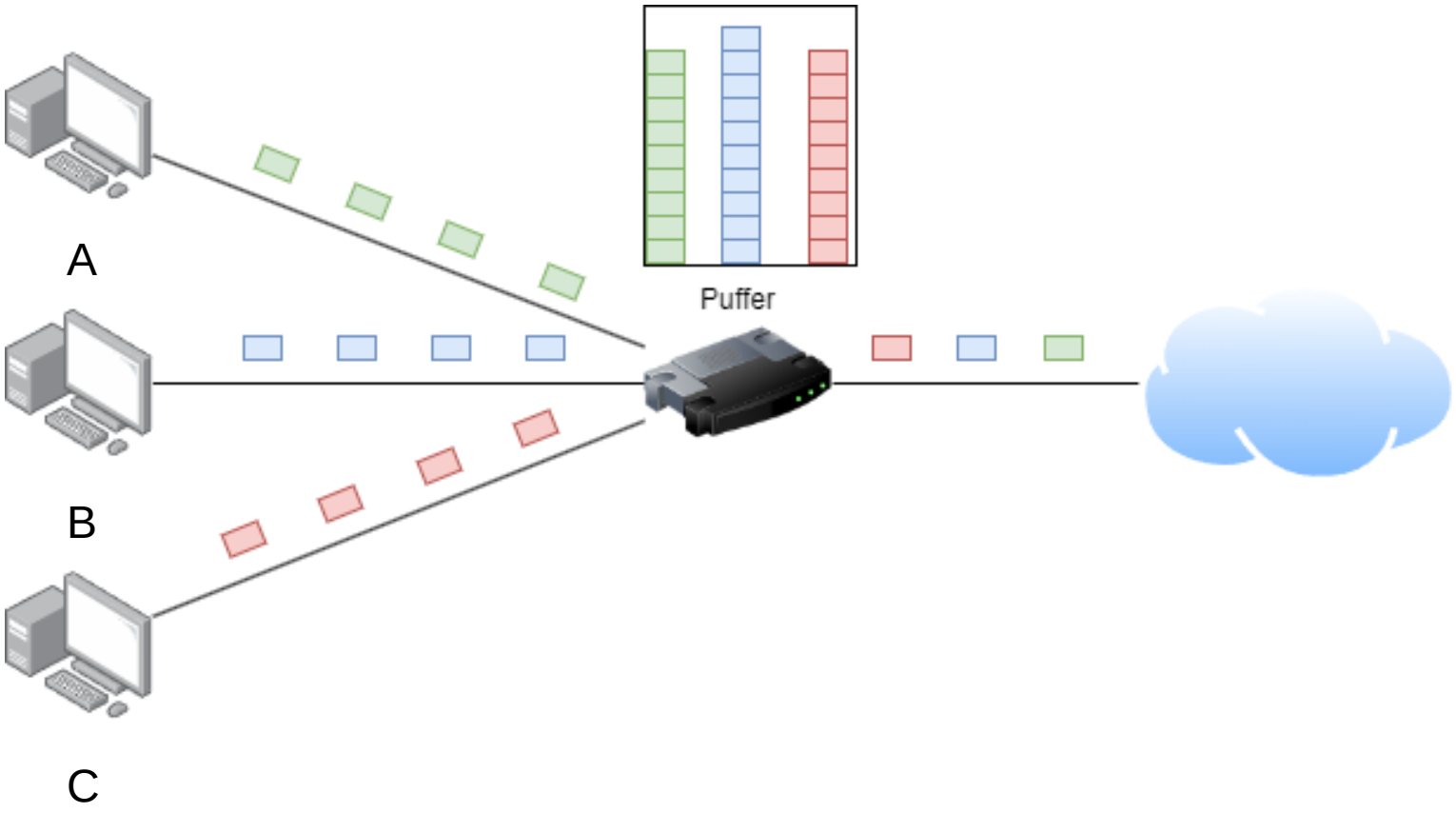




Fair Queuing Controlled Delay

- FQ-CoDel ist ein Pufferverwaltungsalgorithmus der Netzwerklatenzen reduziert und die Gesamtleistung verbessert
- Kombination aus CoDel (Controlled Delay) und Fair Queuing (FQ)
- Fair Queuing
  - Netzwerkverkehr wird in verschiedene Datenströme unterteilt welche separat behandelt werden
- Controlled Delay
  - Pakete werden basierend auf der Verweildauer im Puffer und nicht nur abhängig vom Füllstand verworfen

Fair Queuing



- Router leitet von jeder Warteschlange gleichmäßig Pakete weiter
- Datenströme behindern sich nicht gegenseitig

Überblick

7.1 Einführung

7.2 Motivation

7.3 Zusammenspiel zwischen Überlastkontrolle und adaptiver Warteschlangen-verwaltung

7.4 Überlastkontrolle basierend auf Paketverlust

- CUBIC

7.5 Überlastkontrolle basierendauf Verzögerungszeit

- BBR

7.6 Machinelle Lernverfahren für TCP

7.7 Reinforcement Learning for Dynamic Initial Window

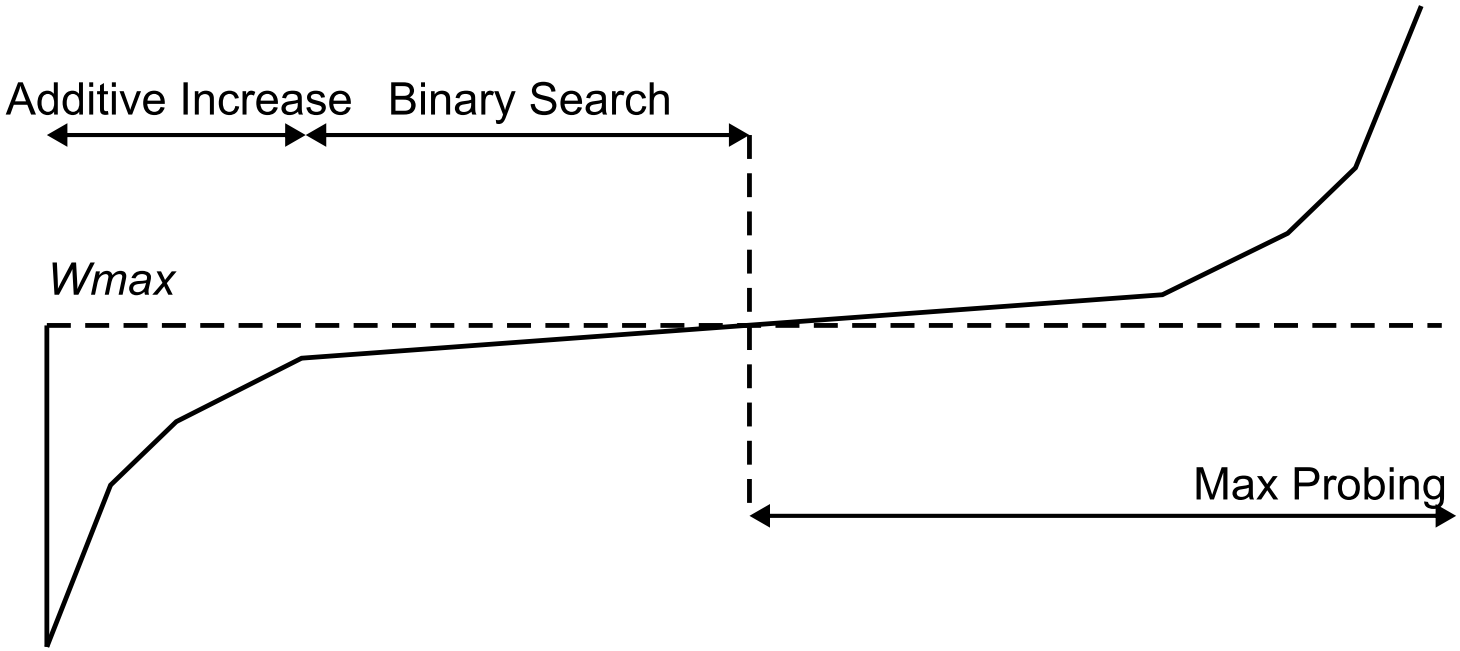
7.8 Reinforcement Learning for Congestion Control

- TCP RL
- Pareto: Fair Congestion Control

TCP BIC

- **B**inary **I**ncrease **C**ongestion Control **BIC**
- Standard im Linux Kernel vor TCP CUBIC
- Hohe Stabilität in „Long Fat Networks“ mit hoher Bandbreite aber auch hoher Latenz
- Schnelle Anpassung an eine veränderte Bandbreite
- Effizienz von TCP BIC kommt durch die Wachstumsfunktion für das Congestion Window (cWnd)

TCP BIC



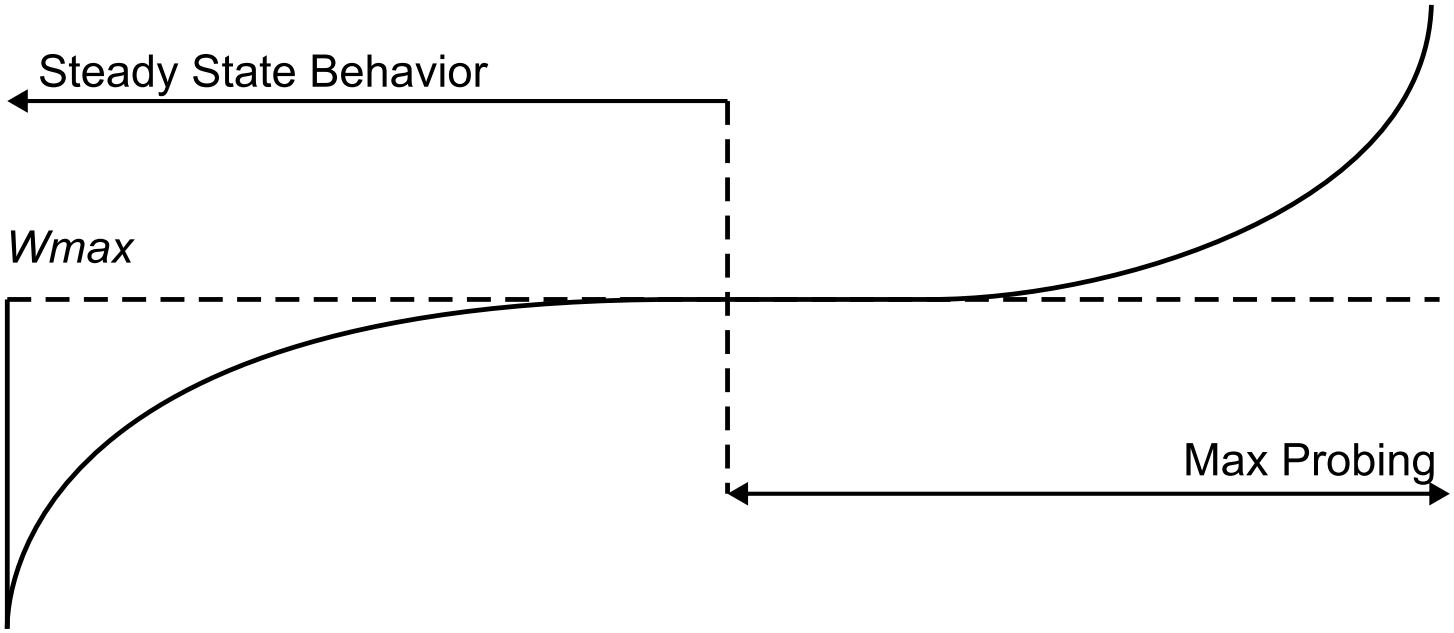
- Anstieg bis zu cWnd vor letztem Paketverlust (W_{max})
- Zunächst Additive Increase bis Differenz zu W_{max} kleiner als Schwellwert, ab dann Binary Search
- Abflachender Anstieg um Wmax, da Paketverlust an dieser Stelle am wahrscheinlichsten ist

TCP CUBIC

- TCP BIC
  - Zu komplex mit vielen Zustände daher schwer zu implementieren
  - Anstieg zu aggressiv besonders dann wenn RTT oder Bandbreite sehr niedrig

- TCP CUBIC ist heute der TCP-Standard in Windows, MacOS und Linux
- Modelliert Wachstumsfunktion von TCP BIC mit einer kubischen Funktion
- Erzielt sehr gute Fairness mit gleichzeitig sendenden Flüssen

TCP CUBIC



- Wachstumsfunktion:
  - $W(t) = C(t-K)^3 + W_{max}$
  - $K = \text{cubic\_root}(W_{max} \cdot \beta / C)$ (Zeit bis $W = W_{max}$)

TCP CUBIC

- Drei Modi

1. TCP-Modus:
   Wenn cWnd kleiner ist als die Fenstergröße, die (standardmäßiges) TCP zum Zeitpunkt t nach dem letzten Paketverlust erreichen würde, dann befindet sich CUBIC im TCP-Modus
2. Konkaver Bereich:
   Wenn cWnd kleiner ist als W_{max}, dann befindet sich CUBIC im konkaven Bereich
3. Konvexer Bereich:
   Wenn cWnd größer ist als W_{max}, dann befindet sich CUBIC im konvexen Bereich.

TCP-Modus

- Bei Empfang eines ACKs wird zuerst überprüft, ob sich CUBIC in der TCP-Region befindet
- Fenstergröße bei TCP mit AIMD
  - $W_{tcp}(t) = W_{max}(1-\beta) + 3(\beta/2-\beta) \cdot t/RTT$

- Wenn cWnd kleiner als $W_{tcp}(t)$ dann befindet ist CUBIC in TCP-Region und cWnd wächst stattdessen mit $W_{tcp}(t)$

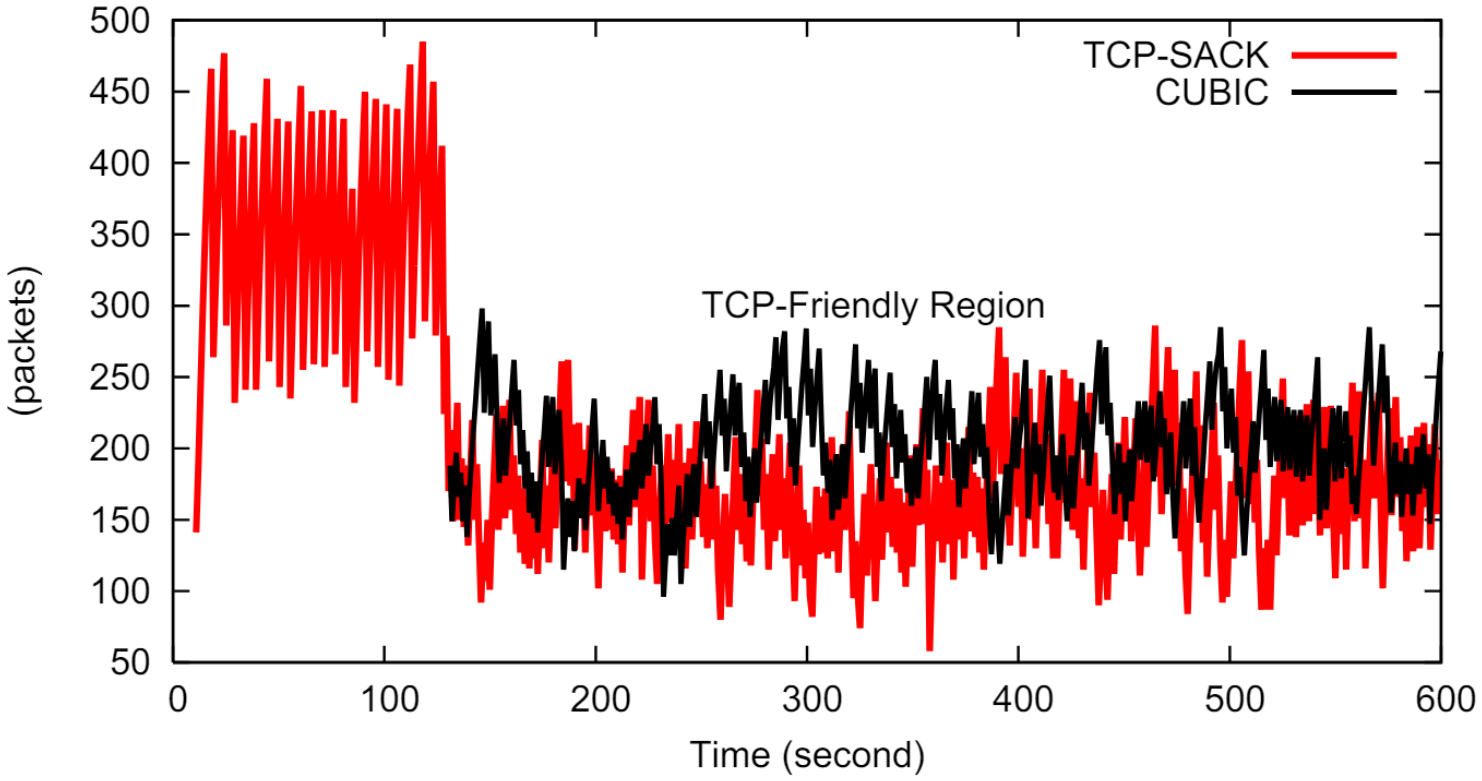
TCP CUBIC

- Wenn TCP CUBIC nicht im TCP-Modus ist dann steigt das cWnd mit jedem Acknowlegment (ACK) um
  - $W_{max}(t + RTT) - cWnd / cWnd$
- Dabei verläuft das cWnd zunächst entlang des konkaven Profils dann rund um W_{max} entlang des Plateaus und zuletzt entlang des konvexen Profiles der Wachstumsfunktion
- Sobald es zu einem Paketverlust kommt wird das cWnd um einen Faktor β verringert

Fast Convergence

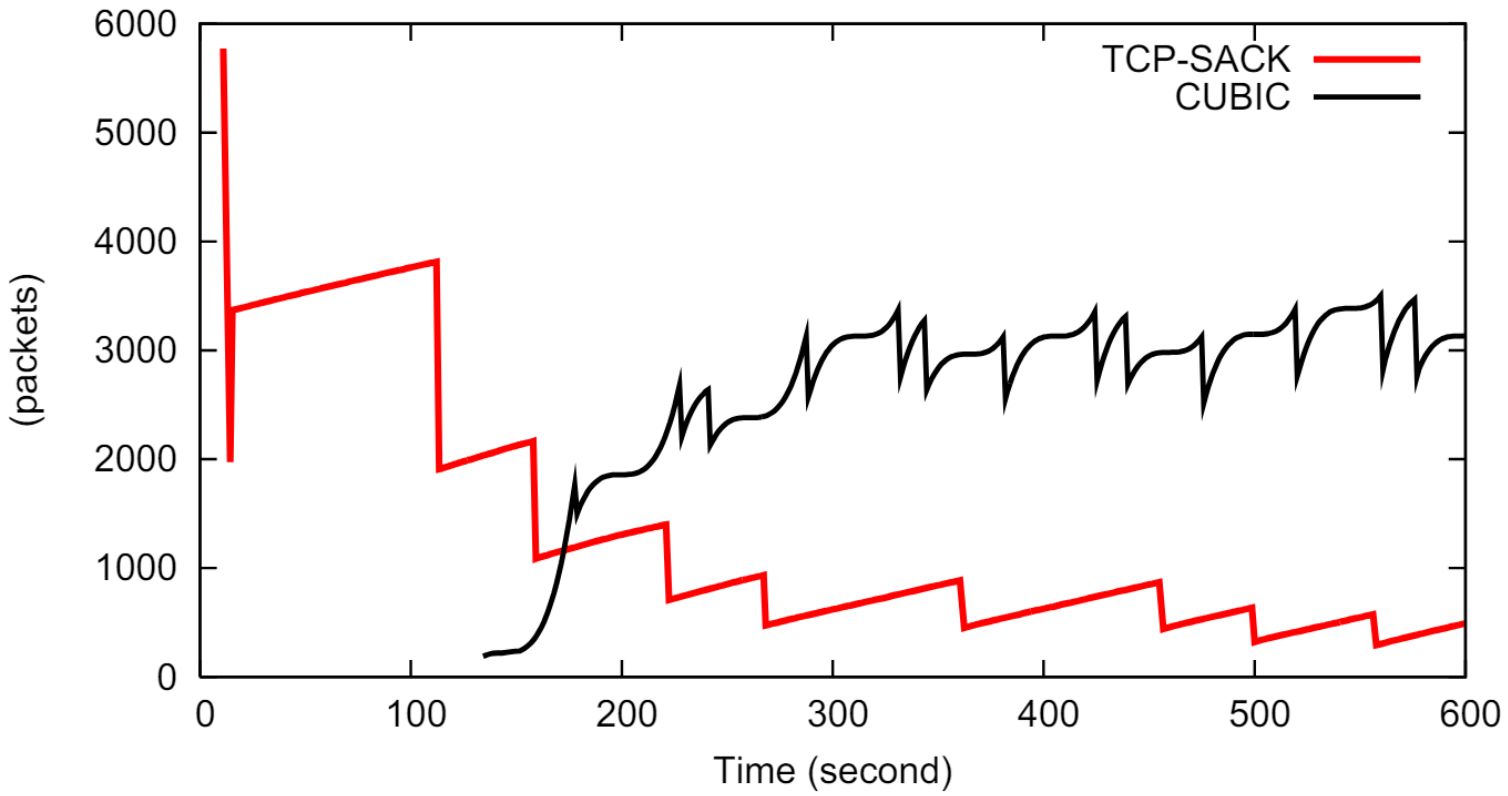
- Heuristik, um sich schneller an veränderte Netzwerkbedingungen anzupassen
- Sobald es zu einem Paketverlust kommt wird das cWnd mit dem vorhergehenden W_{max} verglichen
- Falls das cWnd kleiner ist wird davon ausgegangen, dass ein neuer Datenfluss dem Netzwerk beigetreten ist und es wird mehr Bandbreite freigegeben indem W_{max} weiter verringert wird

Evaluation



- SACK und CUBIC Datenübertragung mit 8ms RTT
- Bei so kleiner RTT kann SACK Bandbreite voll ausnutzen
- CUBIC agiert dabei in TCP-Modus

Evaluation



- Gleiches Experiment mit 82ms RTT
- SACK ist nicht mehr in der Lage Bandbreite auszunutzen
- Wachstumsfunktion von CUBIC erkennbar

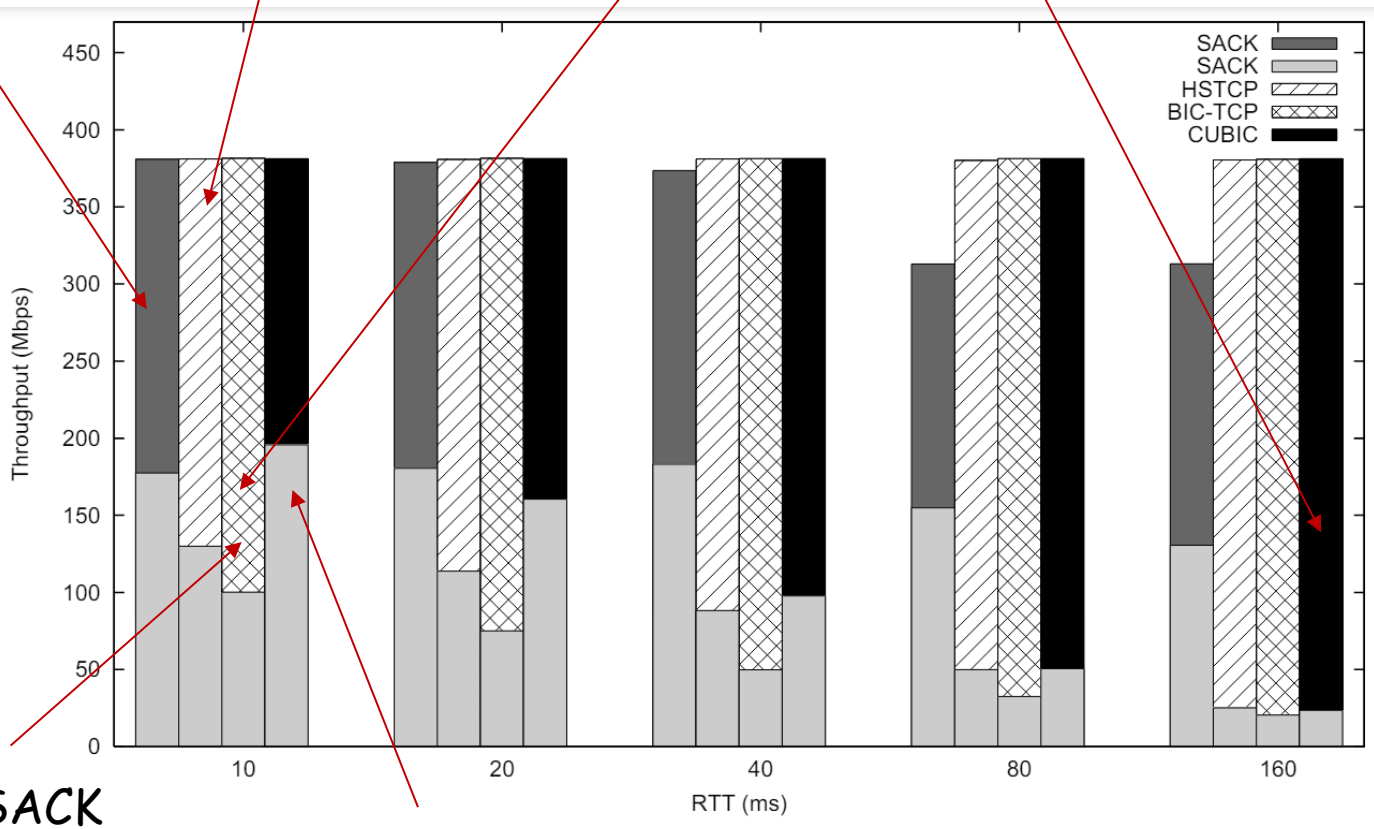
Evaluation

SACK vs SACK

High Speed TCP vs SACK

BIC und HSTCP zu aggressiv

CUBIC nutzt Bandbreite wenn SACK nachlässt



BIC vs SACK

CUBIC vs SACK

Zusammenfassung CUBIC

- Geringere Komplexität
  - Kubische Wachstumsfunktion des Congestion Windows erlaubt einfache Implementierung
- Optimierter Durchsatz
  - TCP-Modus garantiert hohen Durchsatz auch in Netzwerken mit kurzen Round Trip Times
- Verbesserte Fairness
  - Fast Convergence ermöglicht neuen Datenflüssen sehr schnell einen fairen Anteil der Bandbreite

Überblick

7.1 Einführung

7.2 Motivation

7.3 Zusammenspiel zwischen Überlastkontrolle und adaptiver Warteschlangen-verwaltung

7.4 Überlastkontrolle basierend auf Paketverlust

- CUBIC

7.5 Überlastkontrolle basierend auf Verzögerungszeit

- BBR

7.6 Machinelle Lernverfahren für TCP

7.7 Reinforcement Learning for Dynamic Initial Window

7.8 Reinforcement Learning for Congestion Control

- TCP RL
- Pareto: Fair Congestion Control

TCP BBR

- Wiederholung zum Aufblähen des Puffers „Bufferbloat“
  - Verlustbasierte Überlastkontrolle füllt Pufferspeicher vor Engpasskomponenten einer Verbindung und lässt Puffer gefüllt
  - In großen Puffern hängen Pakete besonders lange fest und es kommt zu hohen Latenzen und daraus folgend hoher RTT

- BBR ist eine Alternative zu verlustbasierten Überlastkontrollen

TCP BBR

- TCP BBR wird bereits im WAN von Google verwendet
- TCP BBR wird zum Teil für die google.com Websuche sowie das Abspielen von YouTube Videos genutzt mithilfe von QUIC in HTTP/3
- Weitere Versionen von BBR werden entwickelt
- BBR v2 reagiert zusätzlich auf ECN
- BBR v3 nutzt maschinelle Lernverfahren, um Netzwerkbedingungen zu bestimmen, ist jedoch noch in der Entwicklung

Begriffsklärung

- Die Bandbreite am Flaschenhals (Link mit kleinster Bandbreite auf dem Verbindungspfad)
  - Bottleneck Bandwidth (**B**t**l**B**w**)
- Round Trip Time ohne Störungen oder Verarbeitungszeit
  - Round Trip Propagation Time (**R**Tprop)
- Maximale Menge an Daten, welche über den Verbindungspfad übertragen werden können ohne dass Überlast auftritt
  - Bandwidth Delay Product (BDP = BtlBw · RTprop)

Kleinrocks optimaler Arbeitspunkt

- 1979 hat Leonard Kleinrock gezeigt das eine Senderate gleich dem BDP optimal die zur Verfügung stehende Bandbreite ausnutzt und Verzögerungen, sowie Paket Verluste minimiert

- Jeffrey Jaffe ungefähr zur gleichen Zeit bewiesen, das es nicht möglich ist, ein Algorithmus zu entwerfen, welcher zu diesem Punkt konvergiert

Kleinrocks optimaler Betriebspunkt

- Warum war es nicht möglich der optimalen Betriebspunkt für das BDP zu bestimmen?
- Gründe für den Anstieg der RTT nicht eindeutig
  1. Verbindungspfad wird länger
  2. Bandbreite der Engpasskomponente wird kleiner
  3. Neue Datenübertragungen füllen Puffer im Router

- Lösung
  - Verbindung über eine längere Zeit beobachten

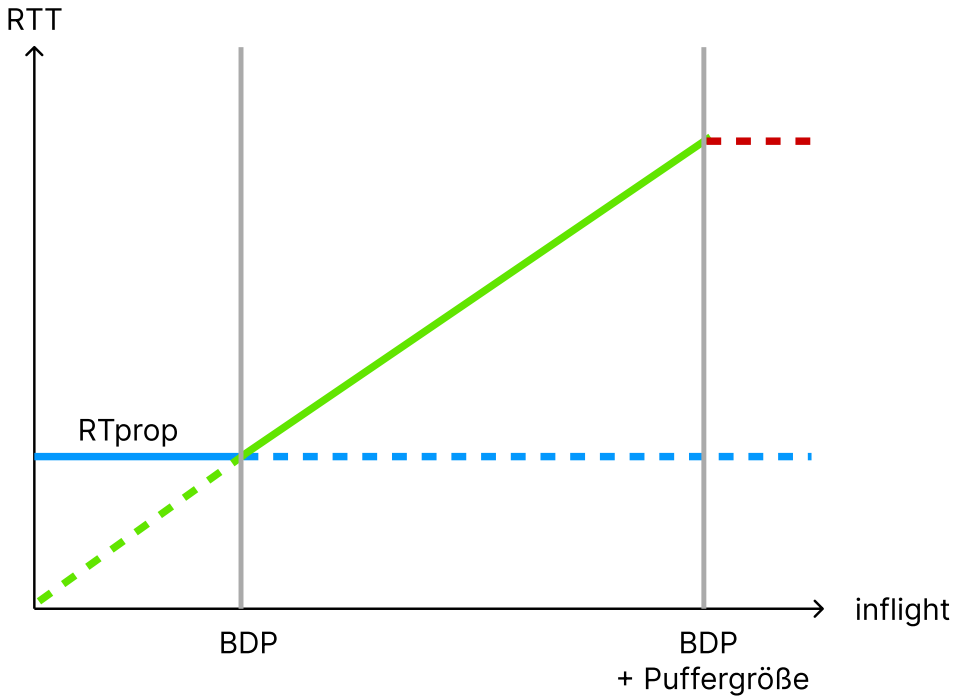
TCP BBR

- Google entwirft Überlastkontrolle und bestimmt dafür
  - Bottleneck Bandwidth, **B**tl**B**w
  - Round-trip propagation time, **R**Tprop

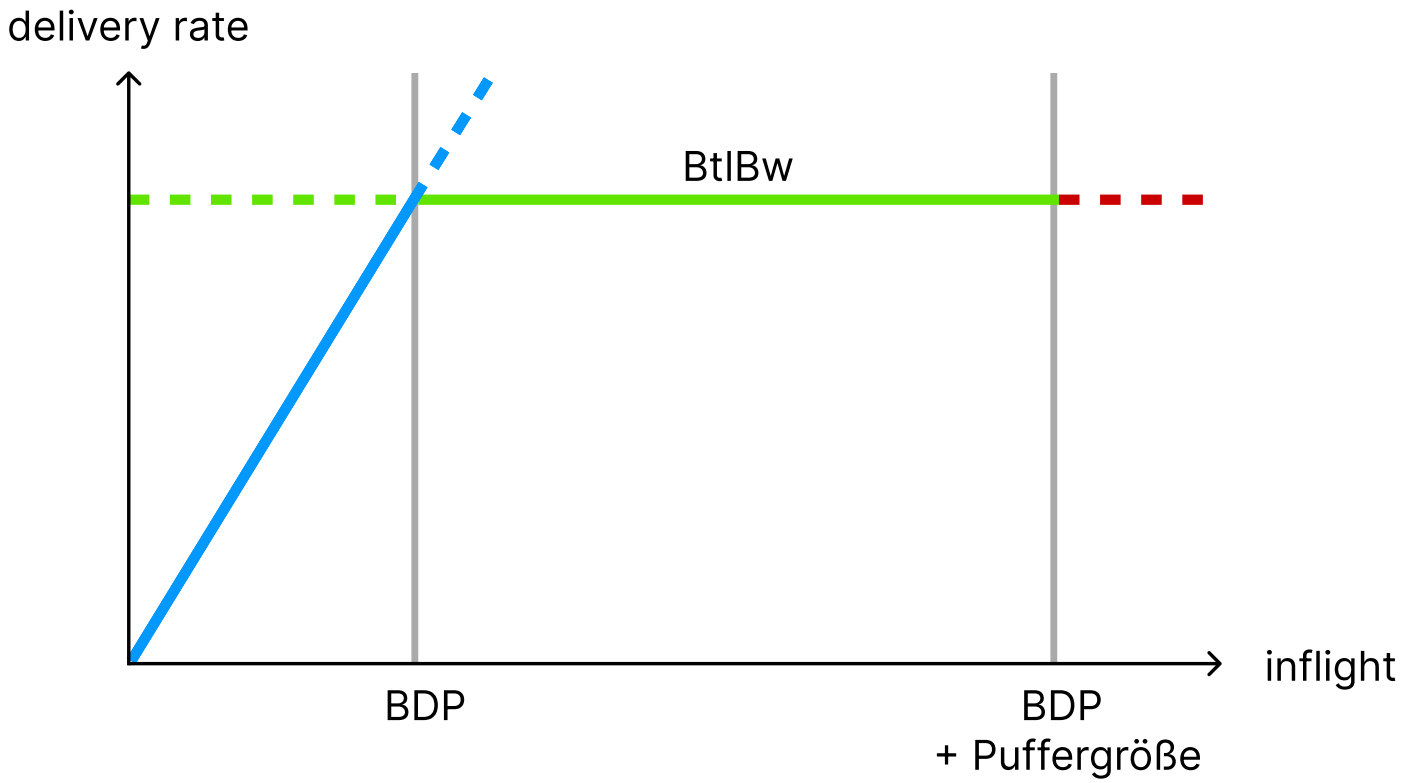
- BBR reagiert auf Überlast anstatt auf Paketverlust und erzielt somit einen hohen Durchsatz, Stabilität und Fairness

TCP BBR

- Anwendung sendet weniger Daten als Bandbreite zulässt
  - RTT = RTprop
  - „app limited“
- Bandbreite ausgelastet
  - Anstieg RTT um 1/BtlBw
  - „bandwidth limited“
- Puffer vollständig gefüllt
  - „buffer limited“



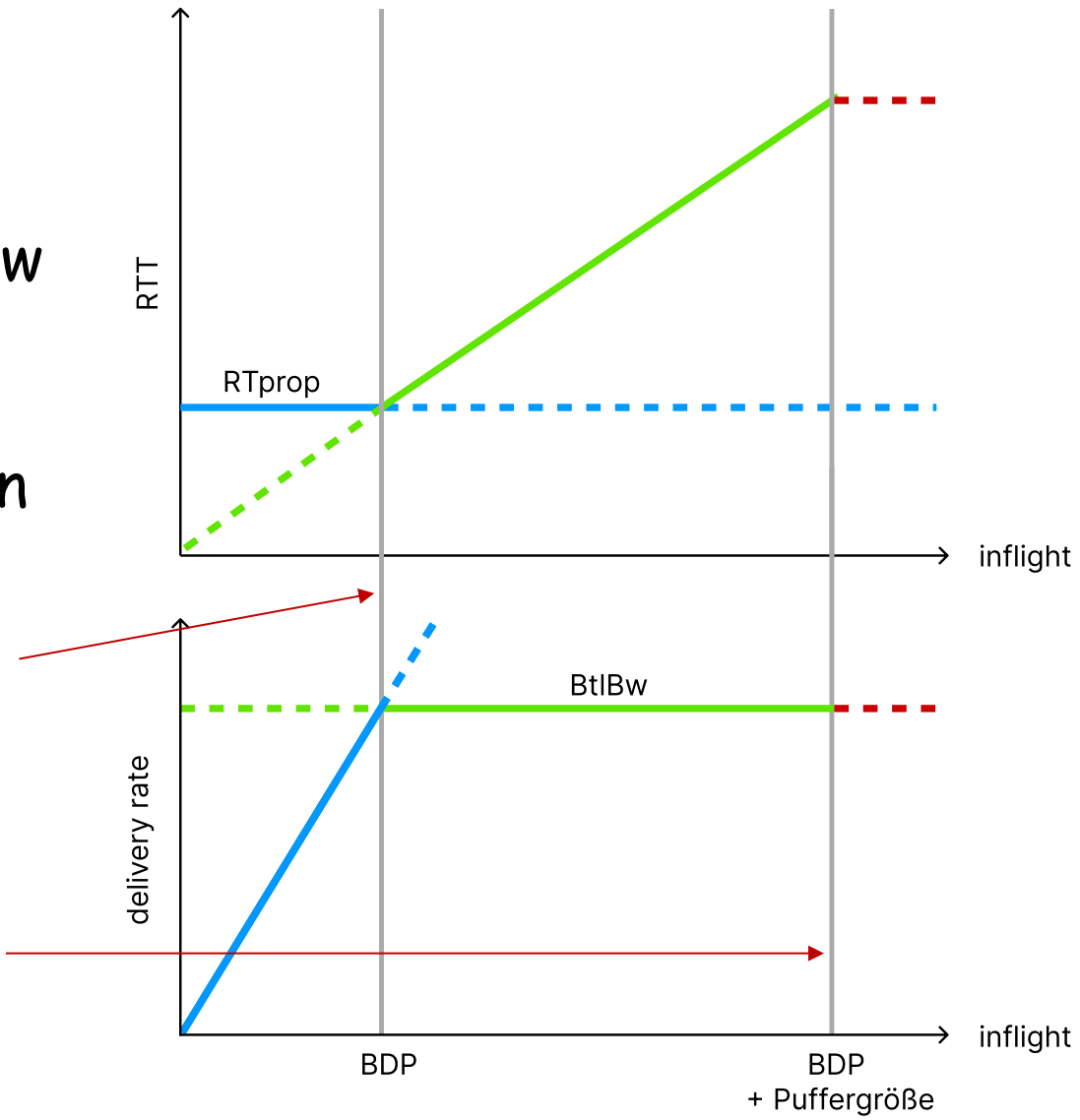
TCP BBR



- Wird die Bandbreite voll ausgenutzt
  - delivery rate = BtlBw

TCP BBR

- RTprop und BtlBw können **nie** zur gleichen Zeit bestimmt werden



TCP BBR

- Bestimmen von RTprop
- Zum Zeitpunkt t ist RTT_t = RTprop + η_t
- $\eta_t \geq 0$ entspricht Störungen (z.B. Wartezeit im Pufferspeicher der Router, Verarbeitungszeit des Pakets am Empfänger oder Anhäufung von Acknowledgement's)

$$\widehat{RTprop} = RTprop + min(\eta_t) = min(RTTt)$$

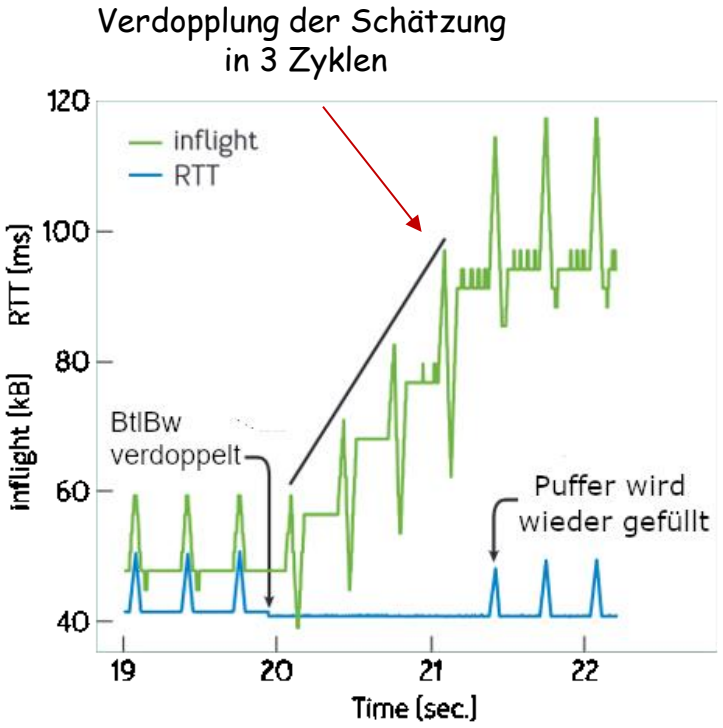
- Bestimmen von BtlBw

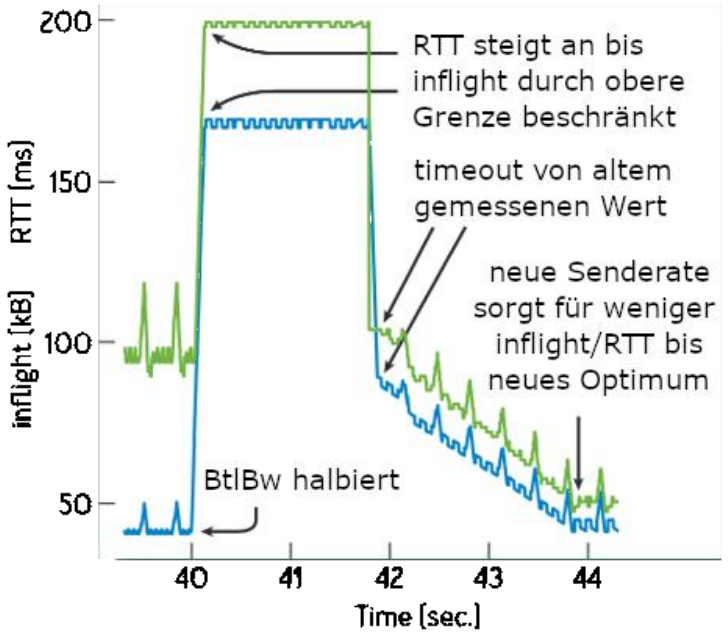
$$\widehat{BtlBw} = max(delivery\ Rate_t)$$

TCP BBR

- Probe Bandwidth „ProbeBW“ Zustand vergleichbar mit Congestion Avoidance
  - Senderate wird in regelmäßigen Zyklen um einen Faktor > 1 angehoben, um neuen Wert für BtlBw zu bestimmen
  - Danach kurzzeitig mit einem Faktor < 1, um mögliche Warteschlangen im Puffer vor dem Flaschenhals abzubauen
- Probe Round Trip Time „ProbeRTT“ Zustand
  - Wenn lange kein neuer minimal Wert (≈RTprob) für RTT gemessen wird Senderate für ein RTT auf vier Pakete reduziert
  - Danach wird in den ursprünglichen Zustand zurückgewechselt
- Gesammelte Werte für RTprob und BtlBw haben Timeout bis diese ungültig werden und zu neuer Schätzung und Senderate führen

TCP BBR



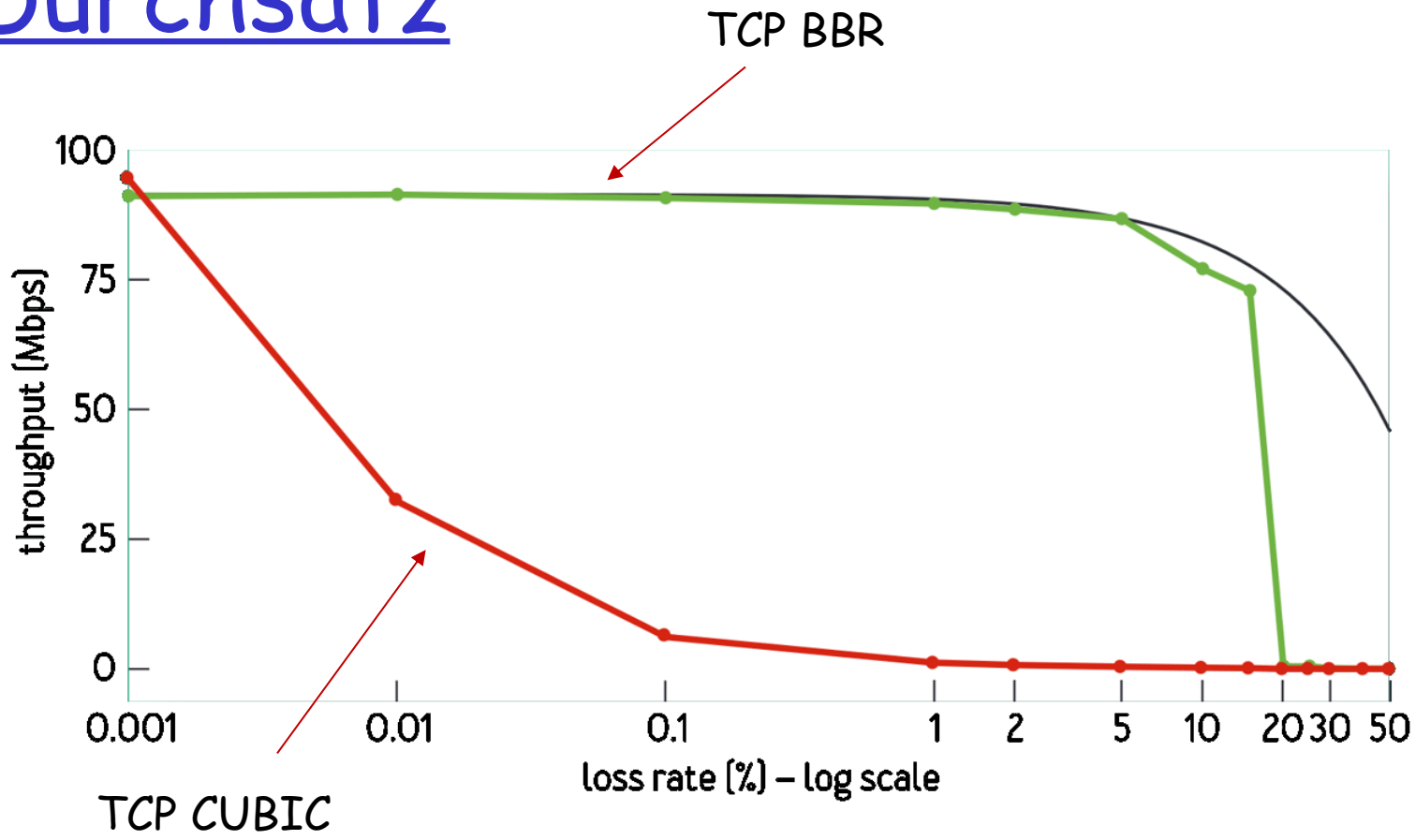


- BtlBw von 10 Mbps verdoppelt sich
- BtlBw von 20 Mbps halbiert sich

Start einer TCP BBR Verbindung

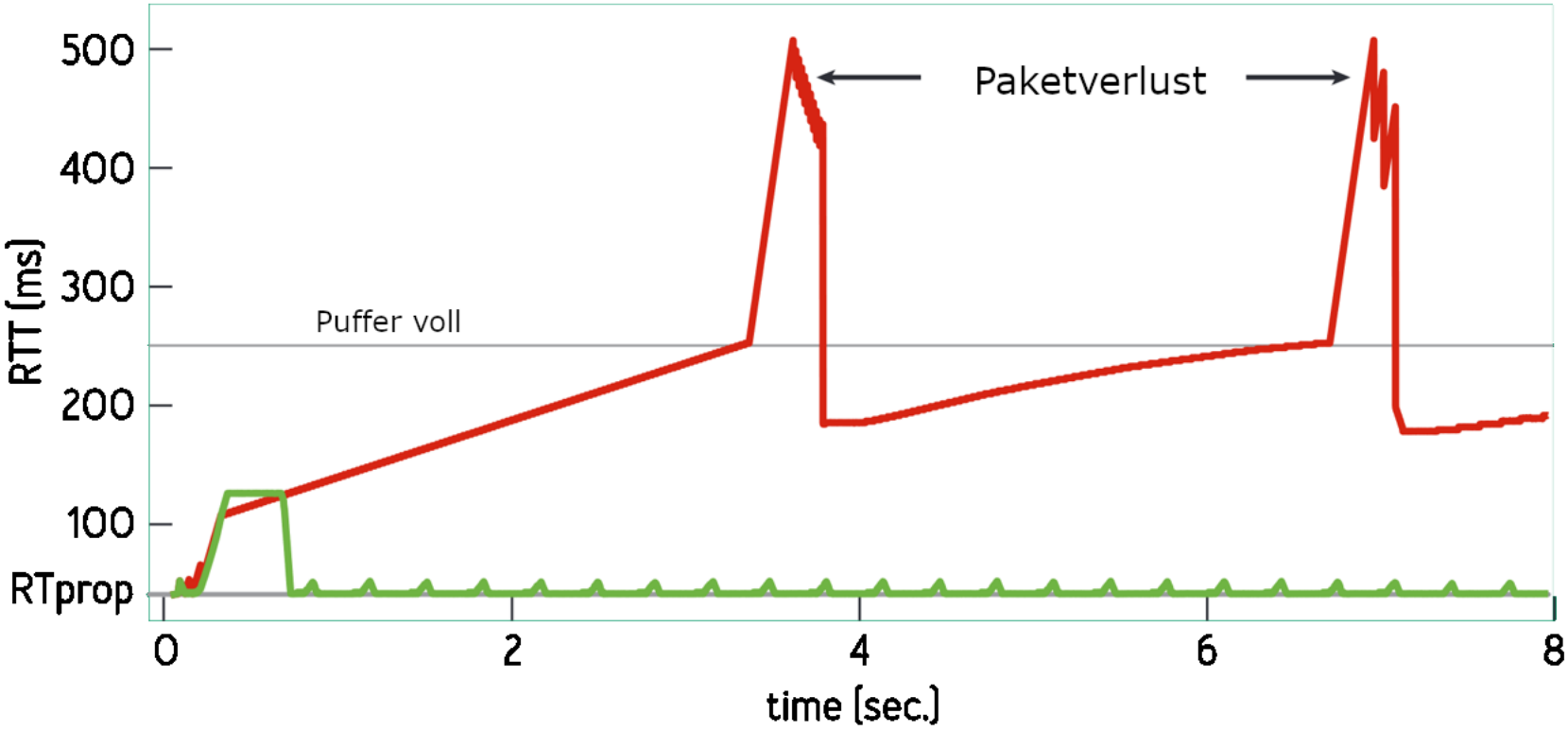
- Startup Zustand
  - Zunächst wird die Senderate mit binärer Suche angehoben, um sehr schnell einen aussagekräftigen Schätzwert für BtlBw zu bestimmen
  - Dabei entsteht jedoch bis zu 2BDP Überhang als Warteschlange im Puffer am Flaschenhals
- Daher nach Startup Wechsel zu Drain Zustand
  - Während Drain wird bewusst inverse von Startup gesendet, um aufgefüllte Puffer wieder zu leeren

Durchsatz



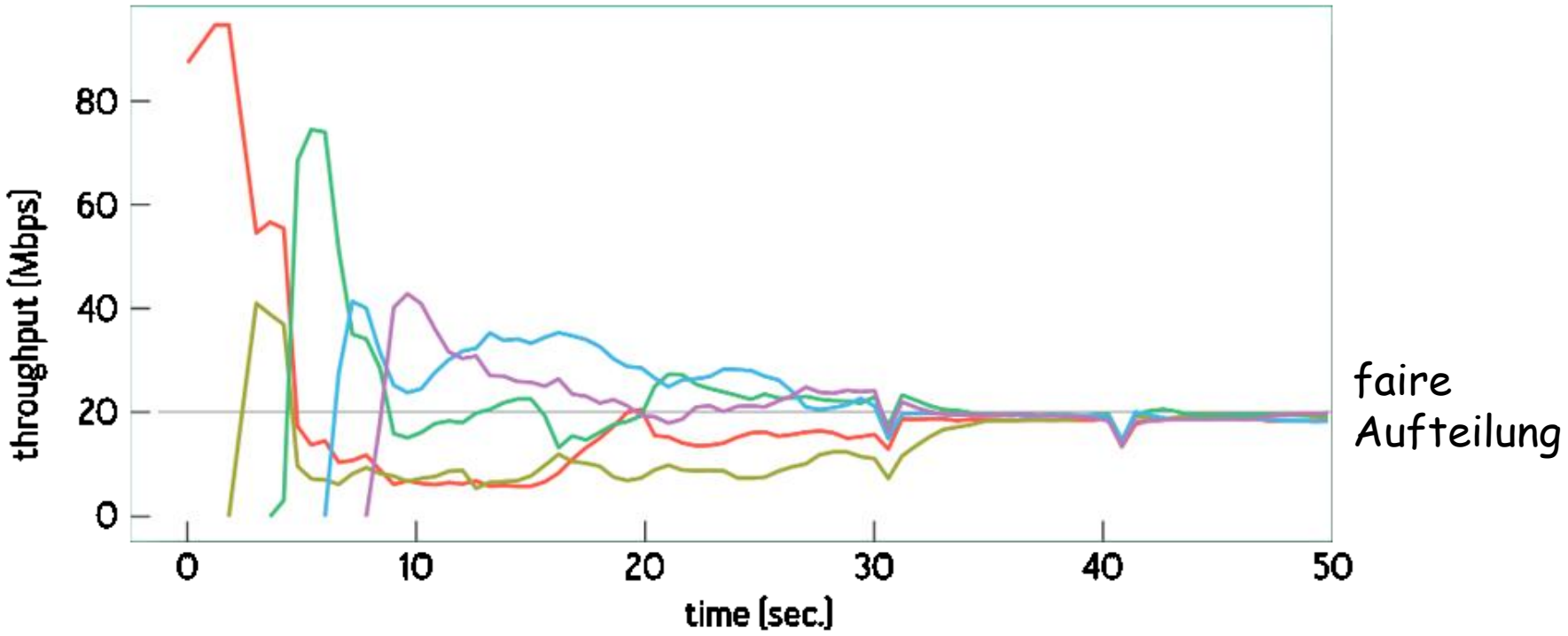
- Bei Verlustwahrscheinlichkeit für Pakete unabhängig von Überlast kleiner Durchsatz für TCP CUBIC

Round Trip Time



- TCP BBR reagiert direkt auf Überlast und füllt somit nie den Puffer vor dem Flaschenhals
- TCP CUBIC Puffer durchschnittlich zu 80% gefüllt

Fairness



- BBR-Datenflüssen gehen in ProbeBW Zustand
- Bandbreite wird neuen Flüssen zur Verfügung gestellt
- Schnelle Konvergenz zu fairem Anteil

Zusammenfassung BBR

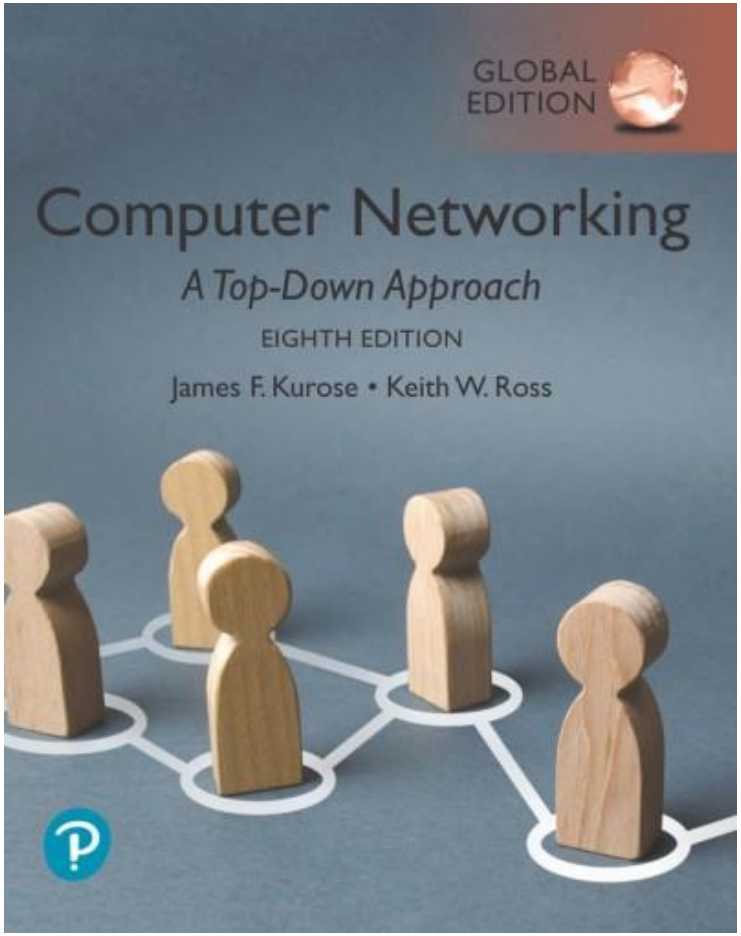
- Optimierte Reaktion auf Überlast
  - BBR bestimmt Netzwerkeigenschaften und kann so direkt auf Überlast statt auf Paketverlust reagieren
- geringere Round Trip Time
  - Es entstehen keine Warteschlagen mehr in den Puffern vor Verbindungsengpässen
- Optimierter Durchsatz
  - In Störanfälligen Netzwerken, erzielt BBR einen sehr hohen Durchsatz während verlustbasierte Überlastkontrollen die Senderate drastisch verringern

Literatur zu TCP CUBIC und BBR

- N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, BBR: Congestion-based Congestion Control, ACM Queue, 14, 20–53, 2016
- C. Augusto Grazia, N. Patriciello, M. Klapez, and M. Casoni, A Cross-Comparison between TCP and AQM Algorithms: which is the best Couple for Congestion Control, Proc. Int. Conf. on Telecommunications, 75-82, 2017
- S. Ha, I. Rhee and L. Xu, CUBIC: A New TCP-Friendly High-Speed TCP Variant, ACM SIGOPS Operating Systems Review, 42, 64-74, 2008
- D. Zeynali, E. Weyulu, S. Fathalli, B. Chandrasekaran, and A. Feldmann, Promises and Potential of BBRv3, Proc. Int. Conf. on Passive and Active Network Measurement, LNCS 14538, 249–272, Springer 2024

Weiterführendes Lehrbuch zur Vorlesung

James Kurose, Keith Ross, Computer Networking: A Top-Down Approach, Global Edition, 8. Auflage, Pearson, 2021.



Selbststudium

Zum Vertiefen der Inhalte dieser Vorlesung

Leseaufgabe zum Selbststudium bis 7.2.2025:

James Kurose, Keith Ross, Computer Networking: A Top-Down Approach, Pearson, 2021.

Chapter 3: The Transport Layer.

Klausur

Termin: **27.02.2025, 10:30 Uhr bis 11:30 Uhr**

Ort: **Audimax, Augusteum**

Viel Erfolg!!