Analyse der Arbeitsweise von Überlaststeuerungsalgorithmen der TCP Varianten Tahoe, Reno und New Reno

1. Juli 2012

Christopher Bertels

Waldemar Smirnow

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Ziel dieser Simulation ist die Analyse der Arbeitsweise der Überlaststeuerung (Congestion Avoidance) folgender TCP Varianten: Tahoe, Reno, New Reno und dessen Unterschiede.

Es wird das TCP Protokoll Grundwissen vorausgesetzt.

Simulationsaufbau

Um die Überlaststeuerung zu analysieren wird folgende Netzwerktopologie erstellt:

Ein Server der über einen Router zu einem Client verbunden ist. Die Daten-Transferrate zwischen dem Server und dem Router sowie zwischen dem Router und dem Client ist fest auf 2Mbit/s eingestellt. Die Latenzzeit auf den Wegen soll jeweils 20ms betragen und als Paketwarteschlange auf den simulierten Netzwerkgeräten soll Drop Tail verwendet werden.

Der Server soll über das TCP Protokoll Daten an den Client schicken. Es werden ausgewählte Pakete verworfen um Überlast zu simulieren. Dafür werden für jede TCP Variante zwei Simulationen durchgeführt. Einmal wird das elfte Paket verworfen und zum anderen mehrere Pakete in Folge (11, 16, 22).

Dabei werden folgende Daten für die Analyse mitgeschrieben:

- Jedes Paket, so wie es der Server gesendet hat
- Jedes Paket, so wie es der Server empfangen hat
- Die congestion window size (cwnd) auf dem Server
- Der slow start threshold (ssthresh) auf dem Server

Simulationsdurchführung

Für die Durchführung der Simulation wird das Netzwerk Simulationsframework ns3 verwendet. Dabei werden drei Nodes (Server, Router und Client) erstellt. Der Server ist über das PointToPoint NetDevice mit dem Router verbunden und befindet sich in dem 10.0.0.0/8 Netzwerk. Der Client ist ebenfalls mit dem Router über ein weiteres PointToPoint NetDevice verbunden und ist in dem 192.168.178.0/24 Netzwerk. Der Router soll die beiden Netzwerke koppeln (eine Route herstellen). Auf dem Client läuft eine PacketSink Applikation (auf dem Port 55000), die als Aufgabe den Empfang von TCP Daten hat. Der Router schickt mittels einer Hilfsanwendung (aus dem ns3 Tutorial) TCP Daten an den Client. Alle Pakete, die den Server verlassen werden mit Hilfe eines Callbacks in die Datei „plotdata/server-trace-tx.data“ protokolliert und die ausgehenden Pakete in „plotdata/server-trace-rx.data“. Änderungen an cwnd und ssthresh werden in die Datei „plotdata/server-cw-ssth.data“ mitgeschrieben. Der Server soll in der zweiten Simulationssekunde anfangen die Daten zu übertragen und in der neunten keine neuen Daten mehr generieren.

Die gesammelten Daten werden mittels gnuplot zu Diagrammen aufbereitet, die im Verzeichnis „plotdata/<TCP Variante>/11(-16-22)“ zu finden sind. Für die Analyse werden gesonderte Diagramme für die zweite bis vierte Sekunde erstellt (Paketverlust Zeitraum).

Unterschiede der TCP Varianten bei der Überlaststeuerung

Überlaststeuerung Allgemein (Congestion Avoidance)

Slow-Start

Zu Beginn einer Übertragung wird der Slow-Start Algorithmus verwendet, um ein sog. "Congestion Window" (Überlastfenster) zu bestimmen. Dies entspricht der Anzahl an Bytes, die vom Sender verschickt wurden, aber noch nicht vom Empfänger bestätigt (ACKed) wurden. Dieser Wert wird vom Sender verwaltet (geschätzt). Zu Beginn wird dieser Wert auf 2 MSS (TCP Maximum Segment Size) gesetzt.

Der Algorithmus funktioniert so, dass zu Beginn der Übertragung der Congestion Window Wert niedrig steht und dann mit jedem erfolgreichen, vom Empfänger bestätigten (ACKed) Paket, dieser Wert um die Azahl bestätigter Segmente (MSS) erhöht wird. Dies geschieht so lange, bis ein Paket nicht bestätigt wird oder ein festgelegter Schwellenwert ("slow start threshold value") erreicht wird. Danach wächst der Wert linear, der Algorithmus geht über in den Überlaststeuerungsmodus ("congestion avoidance").

Fast Recovery

Es gibt eine weitere Variante des Slow-Start Algorithmus, genannt "Fast Recovery", welcher "Fast Transmit" gefolgt von Überlaststeuerung verwendet. Dieser verhält sich so, dass bei durch 3 aufeinander folgende "duplicate ACKs" signalisierte nicht angekommene Pakete der Überlastfenster-Wert reduziert wird auf den Slow-Start Schwellenwert anstatt auf den Initialwert.

Im Folgenden wird erklärt, was die Hauptunterschiede der verschiedenen TCP Varianten im Umgang mit verlorengegangenen Paketen ("packet loss") sind.

TCP Tahoe

Bei der TCP Variante "Tahoe" wird im Falle eines verlorengegangenen Pakets das Überlastfenster auf 1 gesetzt und mit einem erneuten slow-start begonnen. Dies kann relativ schnell zu einem großen Overhead führen, wenn mehrere Pakete schnell hintereinander verlorengehen. Außerdem werden 3 duplicate ACKs gleich behandelt, wie ein Timeout: Es wird ein "Fast Retransmit" durchgeführt, das Überlastfenster wird auf 1 MSS gesetzt und es wird auf den Slow-Start modus zurückgesetzt.

TCP Reno

Bei der TCP Variante "Reno" werden bei 3 duplicate ACKs das Überlastfenster auf die Hälfte des aktuellen Wertes gesetzt, ein "Fast Retransmit" durchgeführt und geht in den "Fast Recovery" modus über. Im Falle eines ACK Timeouts geht man wie bei "Tahoe" in den Slow-Start modus über.

TCP New Reno

TCP New Reno verbessert die erneute Übermittlung während der "Fast Recovery" Phase, die es auch bei der Variante "Reno" gibt. Für jedes duplicate ACK wird ein noch nicht versendetes Paket am ende des Überlastfensters verschickt. Für jedes ACK das einen partiellen Fortschritt in der Sequenzfolge der zu verschickenden Pakete macht (also nicht immer die gleiche Sequenznummer bestätigt wird sondern die bestätigten Sequenznummern wachsen), wird davon ausgegangen, dass das ACK auf ein neues "Loch" in der Sequenzfolge zeigt (ein weiteres nicht empfangenes Paket). Es wird daraufhin das nächste Paket nach der bestätigten (ACKed) Sequenznummer verschickt.

Analyse

Alle TCP Varianten fangen bis zu dem Zeitpunkt des ersten Paketverlusts gleich an. TCP startet mit dem Drei-Wege-Handschlag (three way handshake) die Verbindung und macht mit Slow-Start weiter. Dies lässt sich durch den exponentiellen Anstieg von *cwnd* und steigender Paketversandzahl pro Zeitschritt feststellen. Unterschiede lassen sich in der Behandlung verlorengegangener Pakete ausmachen. Darauf wird im Folgendem eingegangen. Die Analyse bezieht sich auf den Zeitabschnitt 2-4sec, in dem Paketverlust statt findet. Auf den letzten Seiten sind die Zeitsequenzdiagramme für die ganze Simulationszeit angehängt.

TCP Tahoe

In Abbildung 1 kann man erkennen, dass im Zeitschritt ~2,4 sec der einzige Paketverlust festgestellt wird. Dabei wird cwnd auf 1(MSS) gesetzt wird und ssthresh auf die Hälfte von vorheriger cwnd. In den nächsten Zeitschritten ist ein starker Anstieg von cwnd zu erkennen (exponentiell) bis es ssthresh erreicht. Danach wächst cwnd linear weiter. In Abb. 2 kann man sehen, dass ein Fast-Retransmit statt findet gefolgt von dem Slow-Start und anschließend Congestion Avoidance.

Gehen mehrere Pakete verloren, so behandelt TCP Tahoe den Verlust immer gleich (cwnd := 1MSS, ssthresh:= oldcwnd/2). Es folgt in jedem Fall ein Retransmit und Slow-Start.

Dies entspricht der oben genannten Vorgehensweise von TCP Tahoe.

TCP Reno

In Abb. 3-4 erkennt man gut das Fast-Retransmit (in ~2.5sec) Verhalten bei Paketverlust gefolgt von Congestion Avoidance (ab ~2.6sec). Dabei wird in Vergleich zu TCP Tahoe cwnd unr ssthresh nur auf die Hälfte des alten Wertes herabgesetzt. In Abb. 9-10 sieht man bis zum Zeitpunkt ~2,6sec das gleiche Verhalten, bis weitere Pakete verloren gehen. Nach der kurzen Fast-Recovery Phase folgen zwei Timeouts (begründet durch verlorene Pakete 16 und 22) bei denen cwnd auf 1MSS gesetzt wird und die Slow-Start Phase beginnt. Sobald cwnd den ssthresh überschreitet, wächst cwnd jeweils linear. Man kann die Retransmits zu den Zeitpunkten ~2,9sec und ~3,7sec gut auf Abb. 10 erkennen.

Dies entspricht der oben genannten Vorgehensweise von TCP Reno.

TCP New Reno

Bei alleinigem Paketverlust (Abb. 5-6) lässt sich kein Unterschied zu TCP Reno feststellen.

In den Abb. 11-12 lässt sich eine bessere Behandlung von mehreren aufeinander folgenden Paketverlusten im Vergleich zu TCP Reno feststellen. Insgesamt werden in der gleichen Zeit wesentlich mehr (nahe zu doppelt so viele) Pakete verschickt (man beachte die Y-Skalierung). Cwnd wird nur in einem Fall auf 1MSS gesetzt (Paket 22 in ~3.1sec). Zu der Zeit werden länger keine Pakete verschickt, was auf einen Timeout hin deutet. Anschließend wird mit Slow-Start weiter fortgefahren.

Dies entspricht der oben genannten Vorgehensweise von TCP New Reno.

Fazit

Wie man an den Diagrammen und unserer Analyse gut erkennen kann verbessert TCP Reno das Überlaststeuerungsverhalten bei Paketverlust gegenüber TCP Tahoe da zwischen Timeouts und duplicate ACKs unterschieden wird. TCP New Reno optimiert gegenüber TCP Reno das Verhalten bei duplicate ACKs, was sich auf eine noch bessere Transferrate bei mehreren Paketverlusten auswirkt.

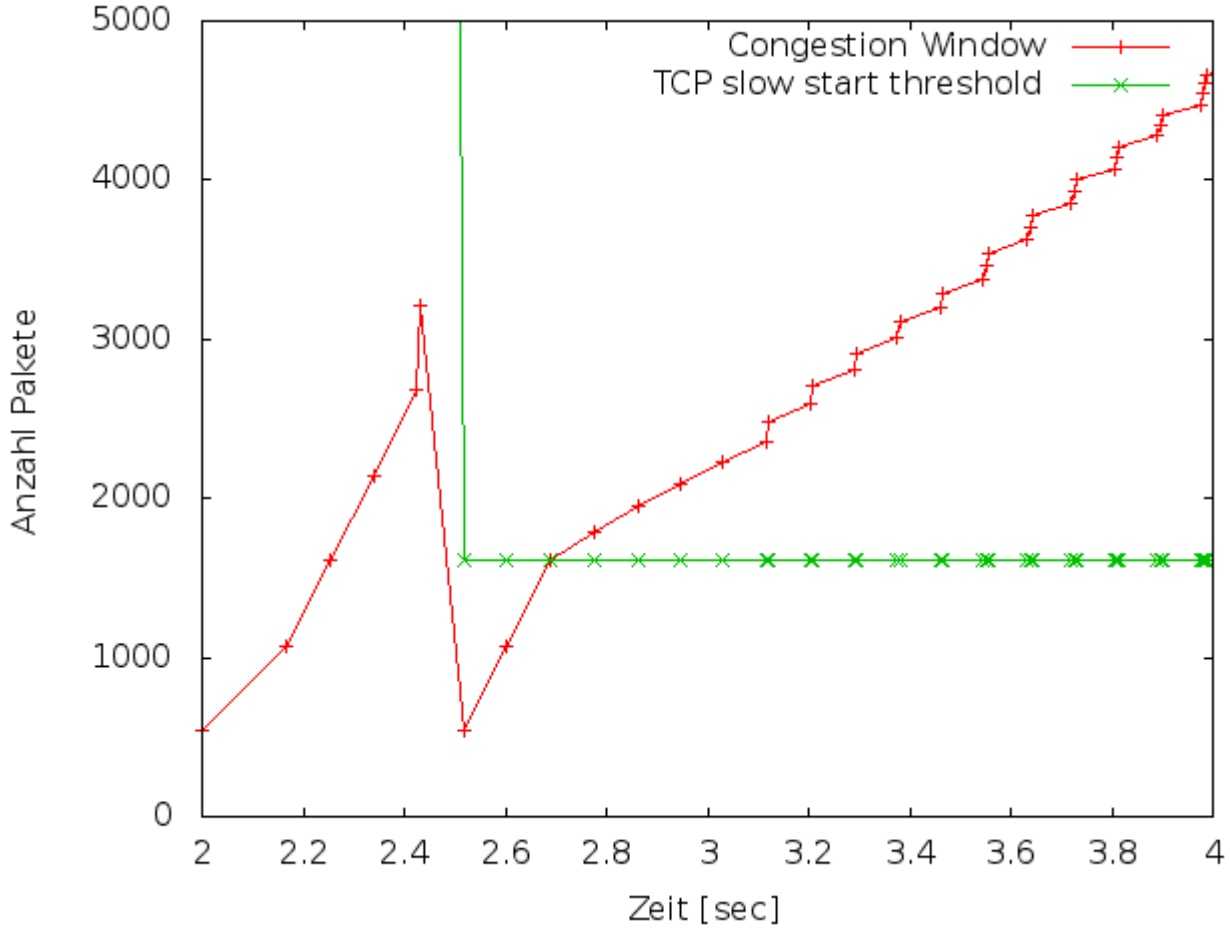

Abbildung 1: TCP Tahoe cwnd und ssthresh gegen die Zeit [Paketverlust: 11]

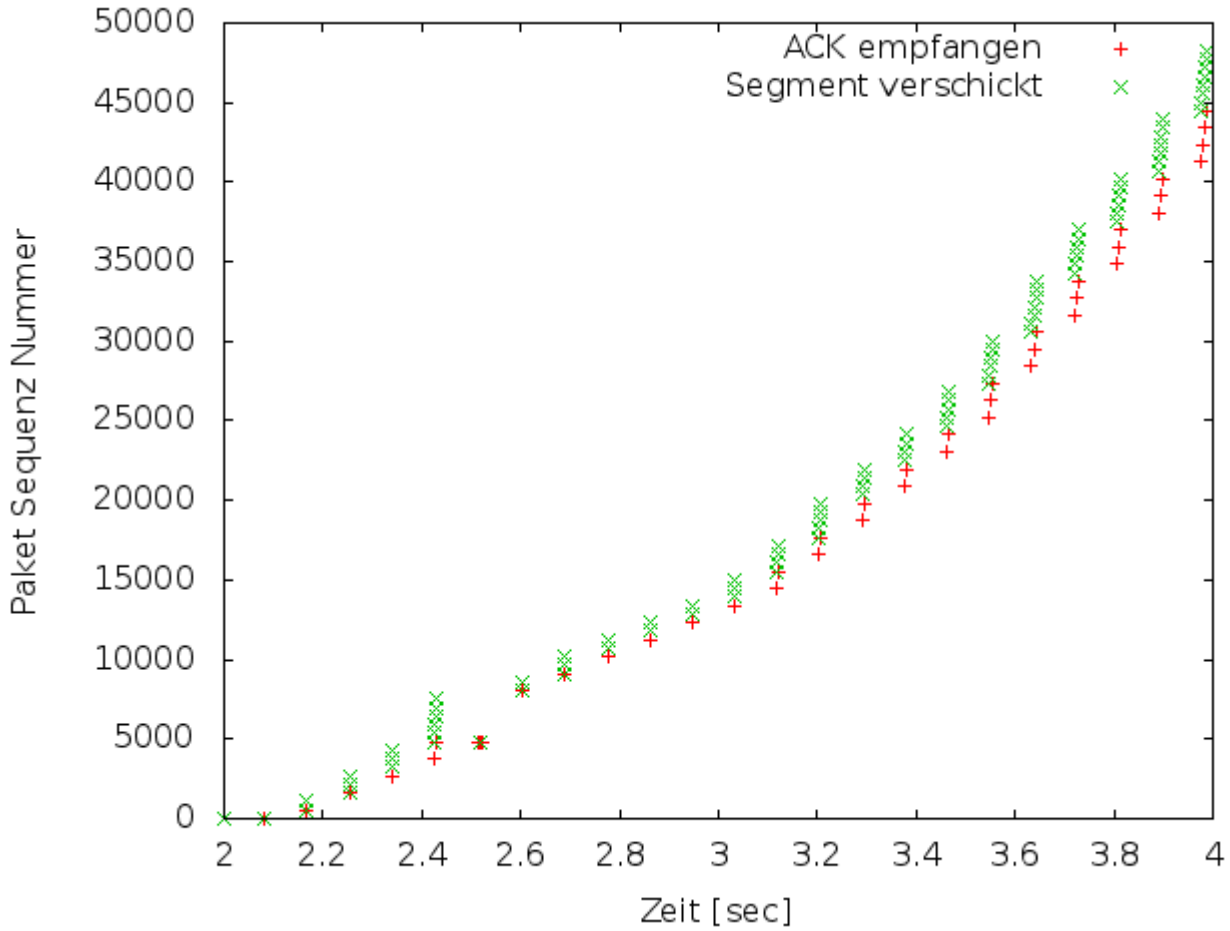

Abbildung 2: TCP Tahoe Zeitsequenzdiagramm [Paketverlust: 11]

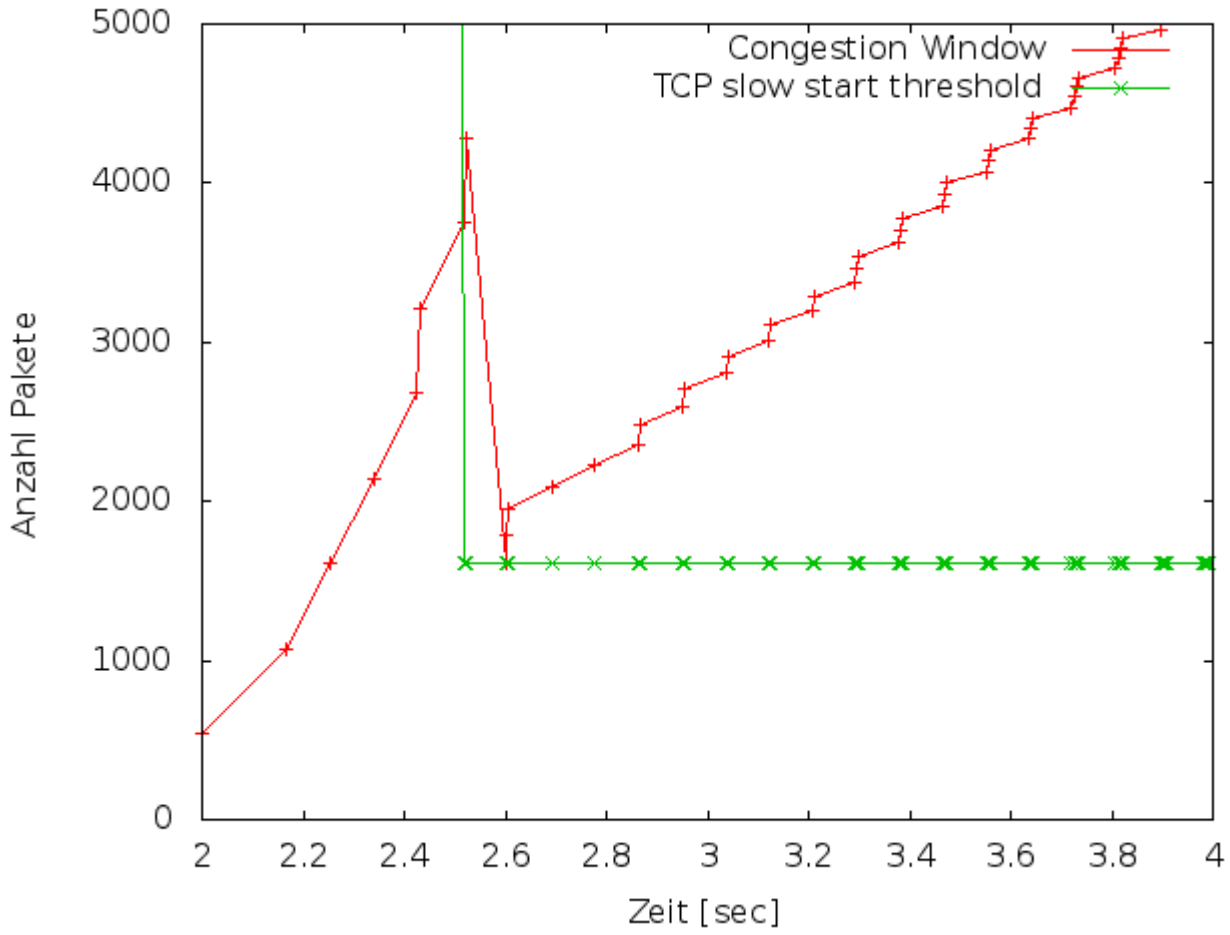

Abbildung 3: TCP Reno cwnd und ssthresh gegen die Zeit [Paketverlust: 11]

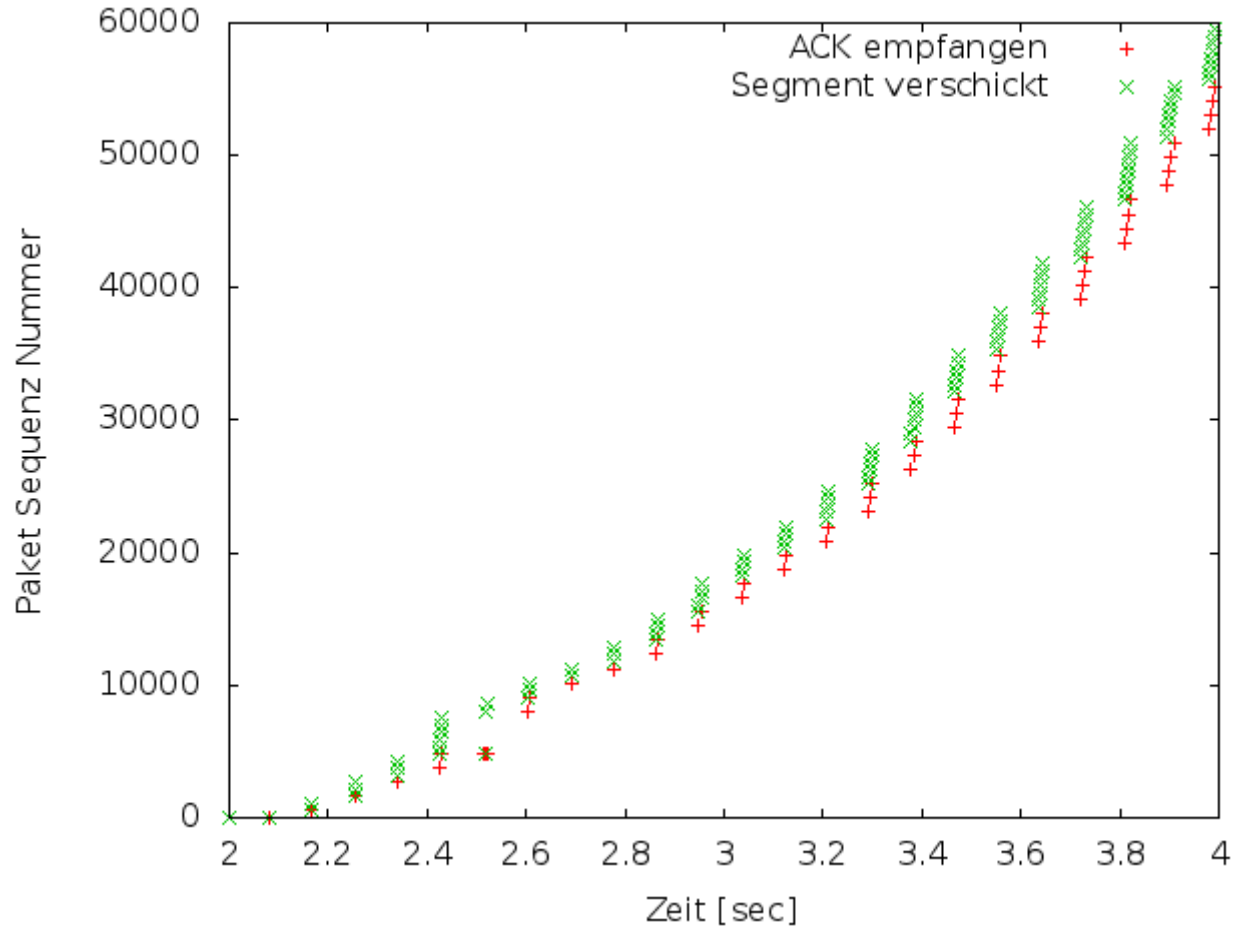

Abbildung 4: TCP Reno Zeitsequenzdiagramm [Paketverlust: 11]

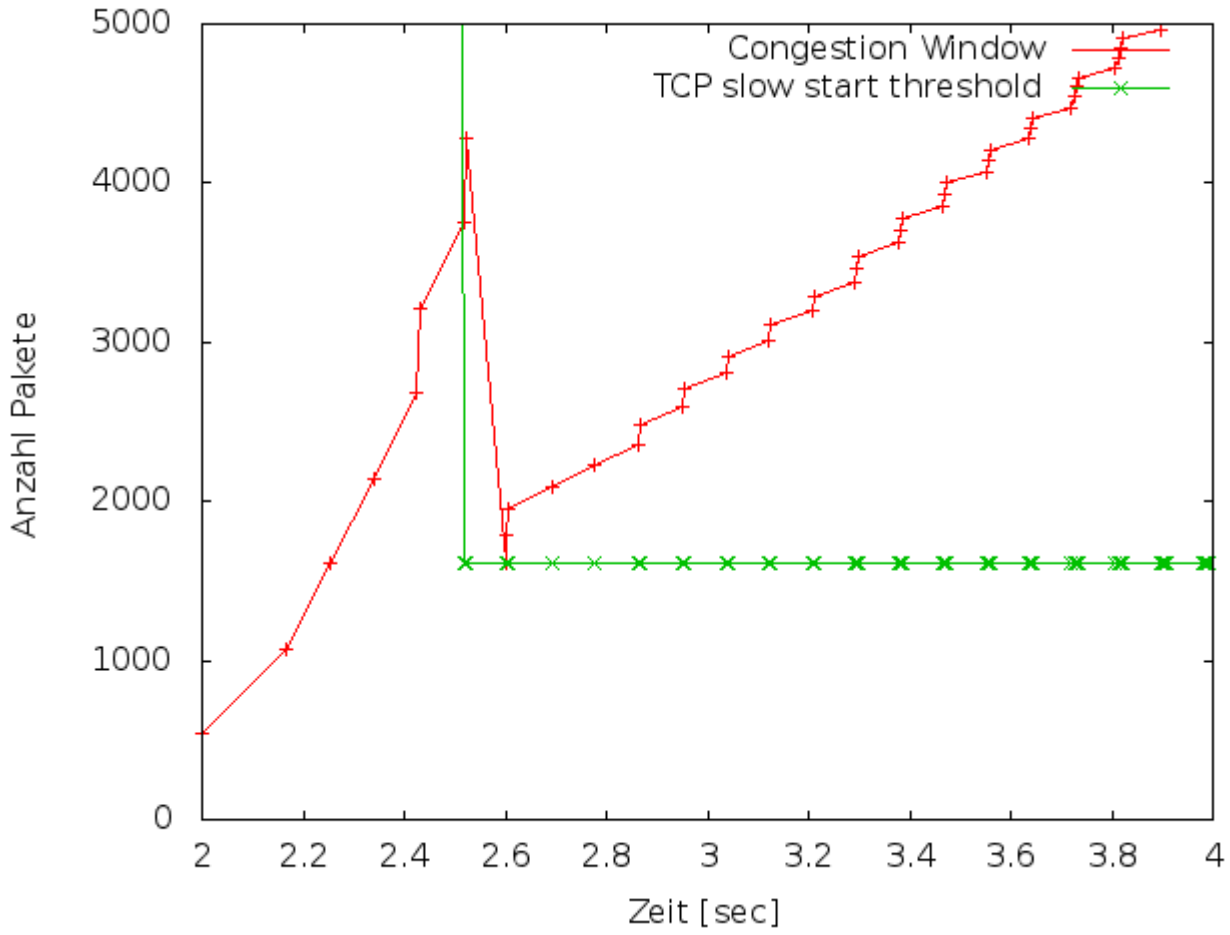

Abbildung 5: TCP New Reno cwnd und ssthresh gegen die Zeit [Paketverlust: 11]

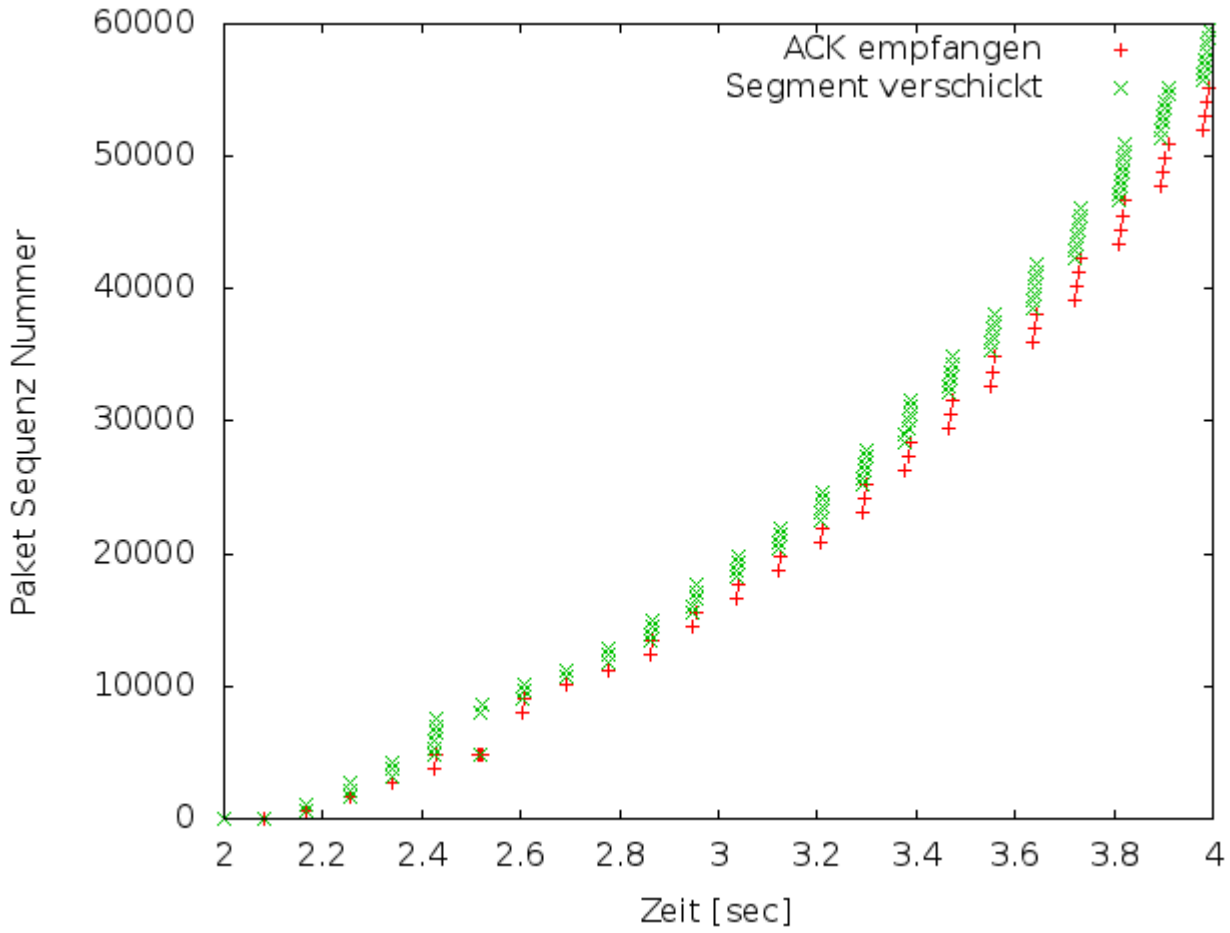

Abbildung 6: TCP New Reno Zeitsequenzdiagramm [Paketverlust: 11]

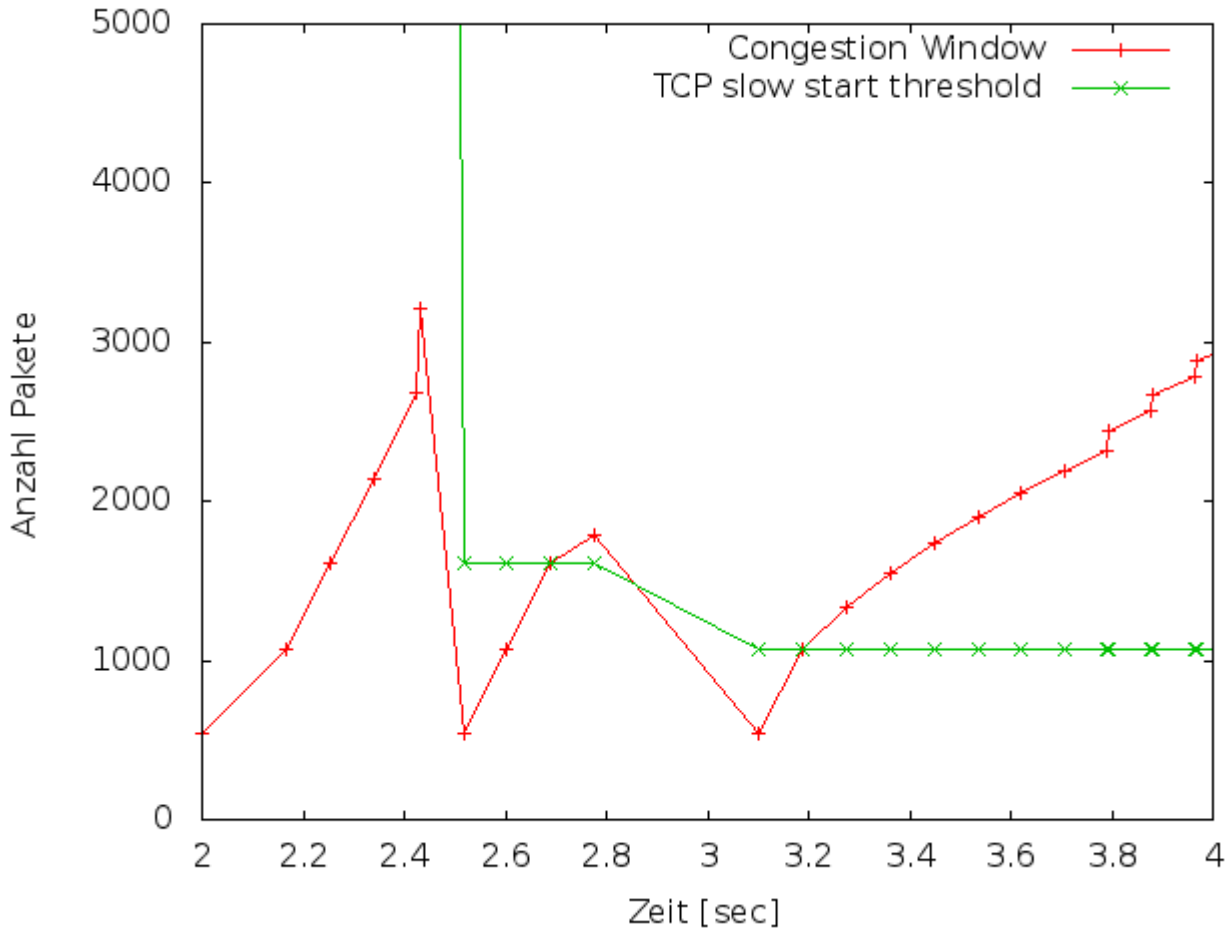

Abbildung 7: TCP Tahoe cwnd und ssthresh gegen die Zeit [Paketverlust: 11, 16, 22]

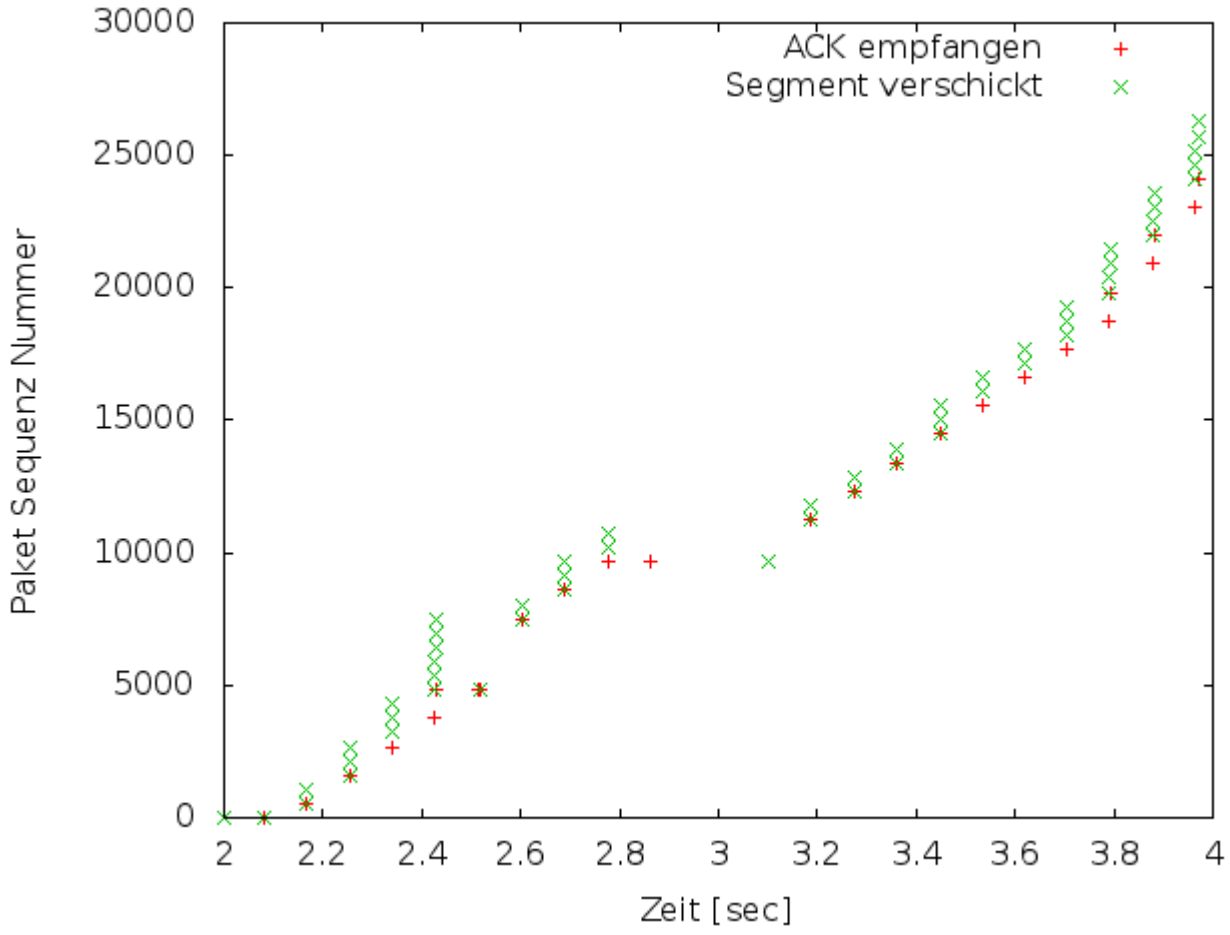

Abbildung 8: TCP Tahoe Zeitsequenzdiagramm [Paketverlust: 11, 16, 22]

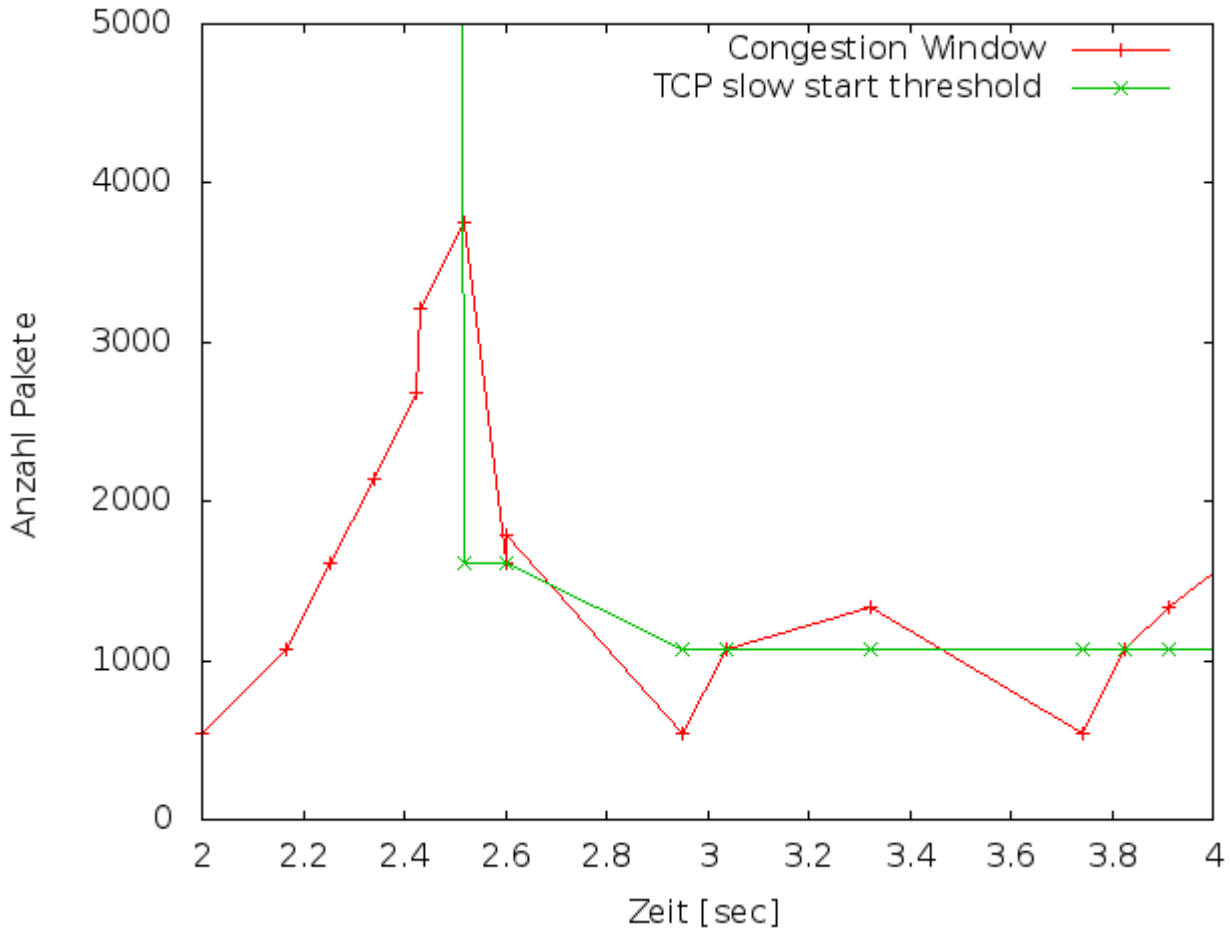

Abbildung 9: TCP Reno cwnd und ssthresh gegen die Zeit [Paketverlust: 11, 16, 22]

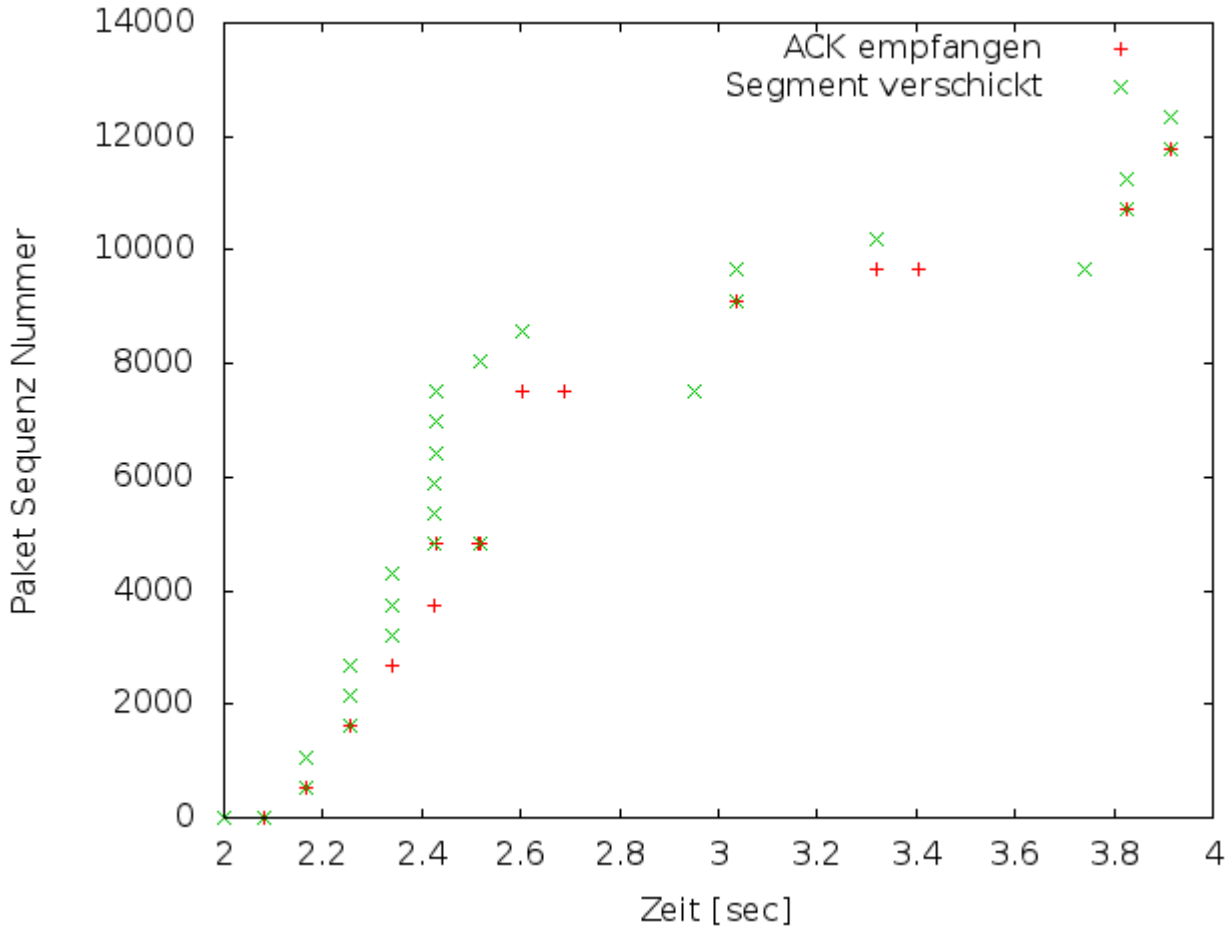

Abbildung 10: TCP Reno Zeitsequenzdiagramm [Paketverlust: 11, 16, 22]

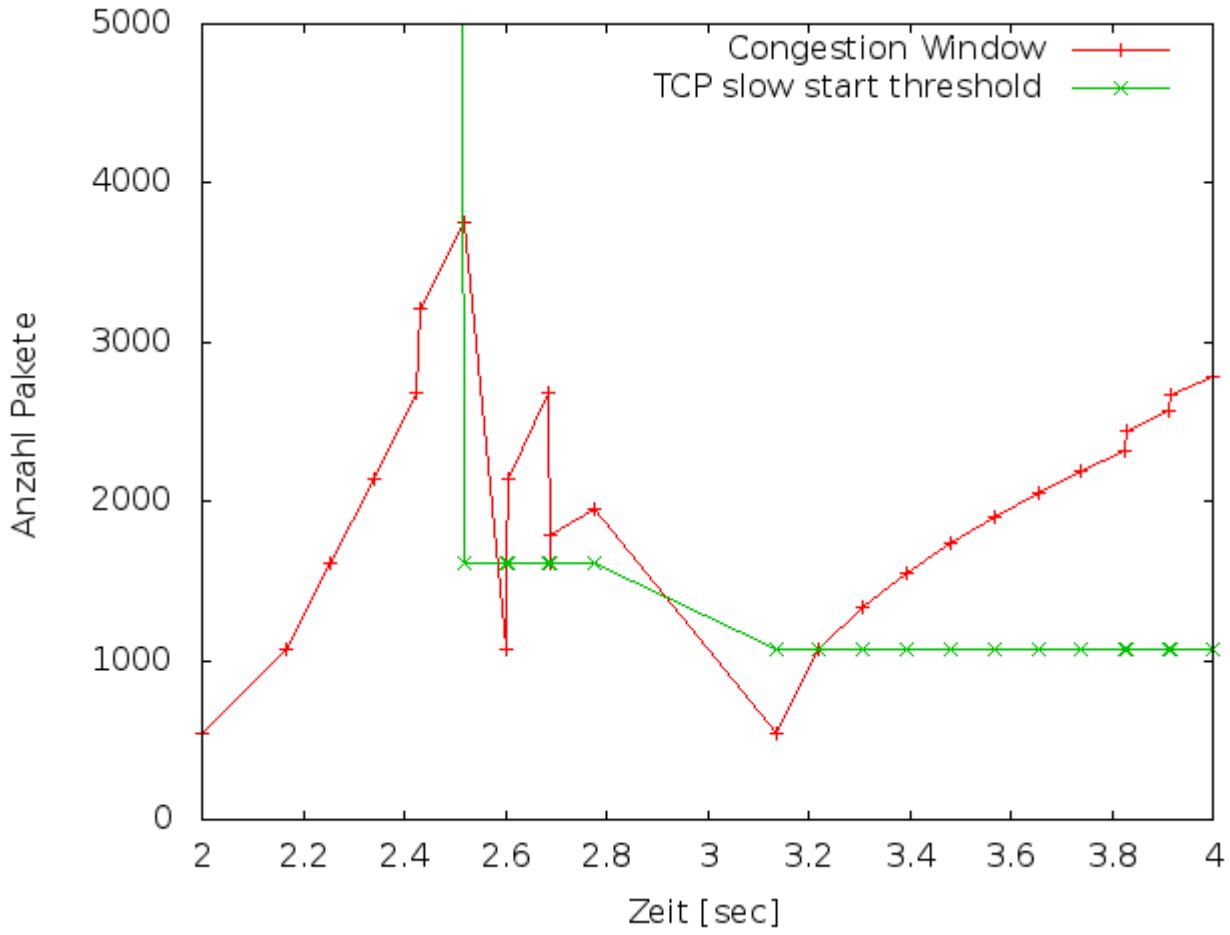

Abbildung 11: TCP New Reno cwnd und ssthresh gegen die Zeit [Paketverlust: 11, 16, 22]

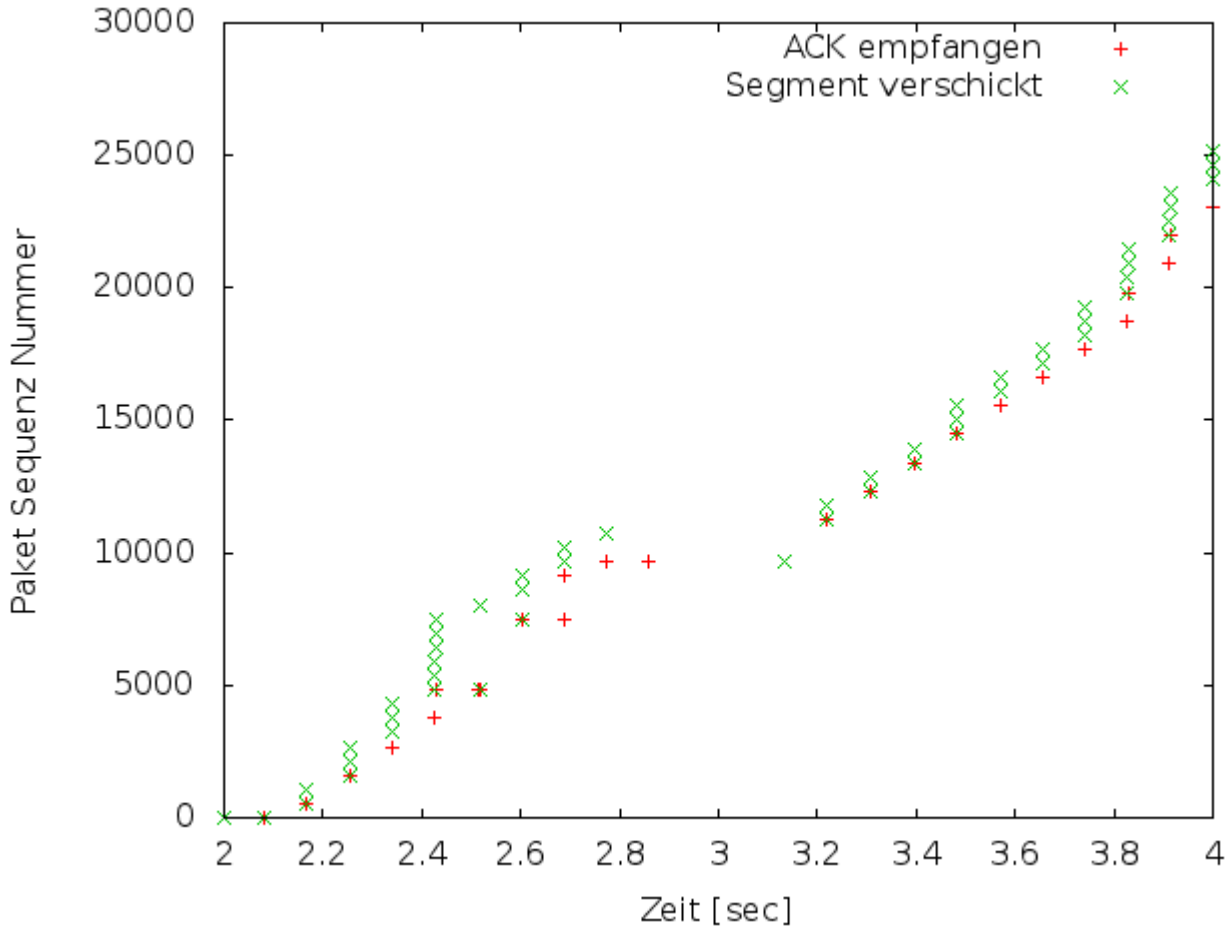

Abbildung 12: TCP New Reno Zeitsequenzdiagramm [Paketverlust: 11, 16, 22]

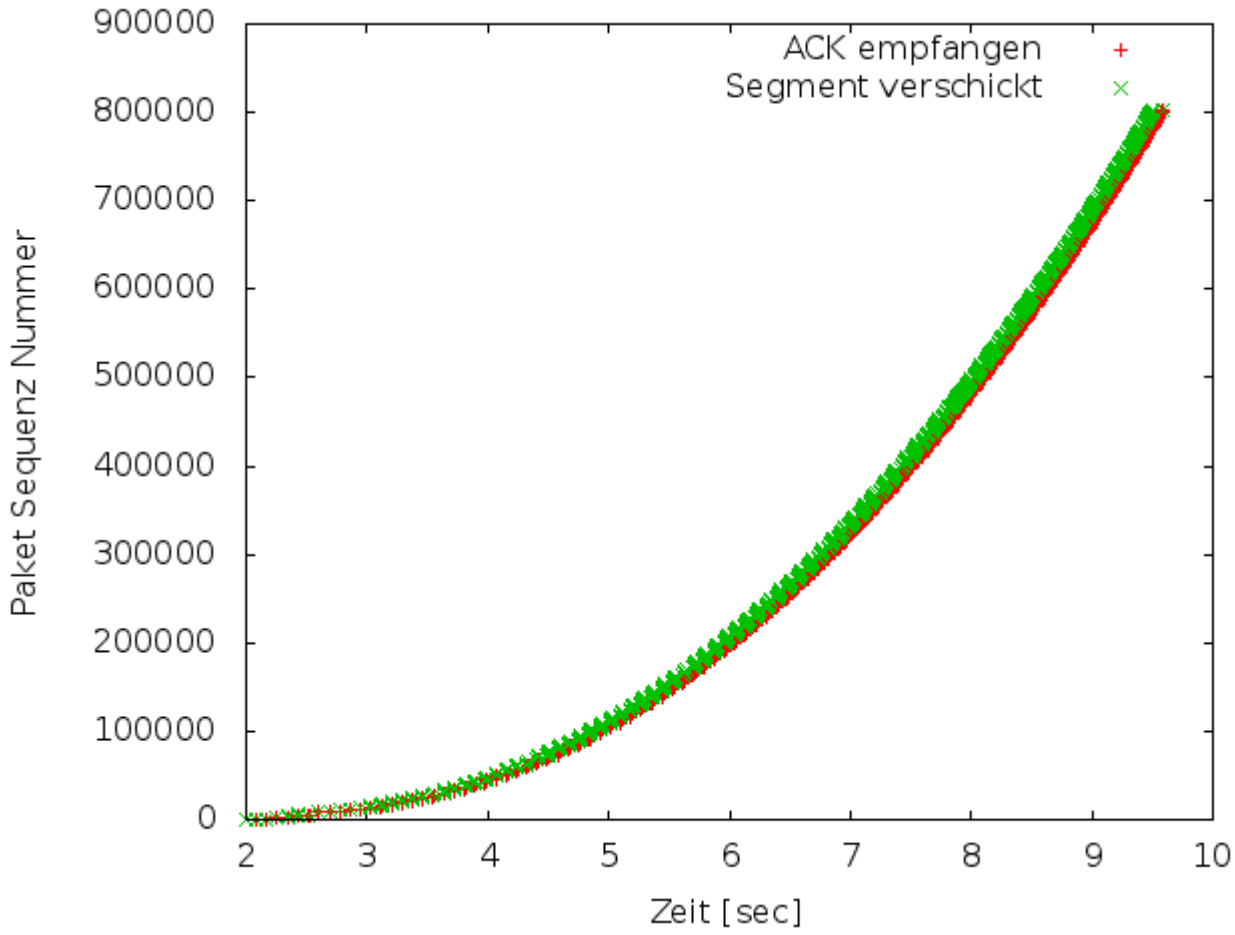

Abbildung 13: TCP Tahoe komplettes Zeitsequenzdiagramm [Paketverlust: 11]

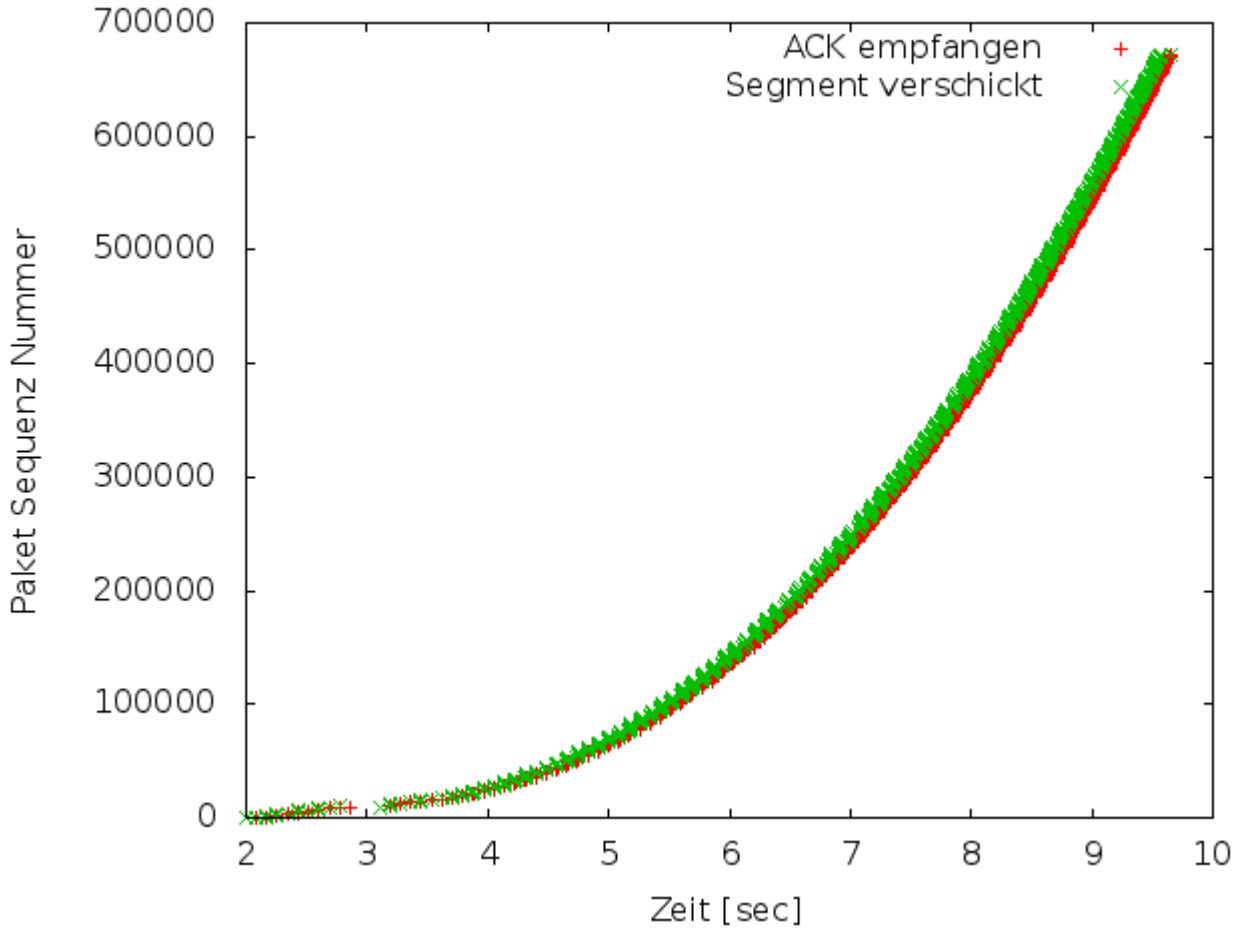

Abbildung 14: TCP Tahoe komplettes Zeitsequenzdiagramm [Paketverlust: 11, 16, 22]

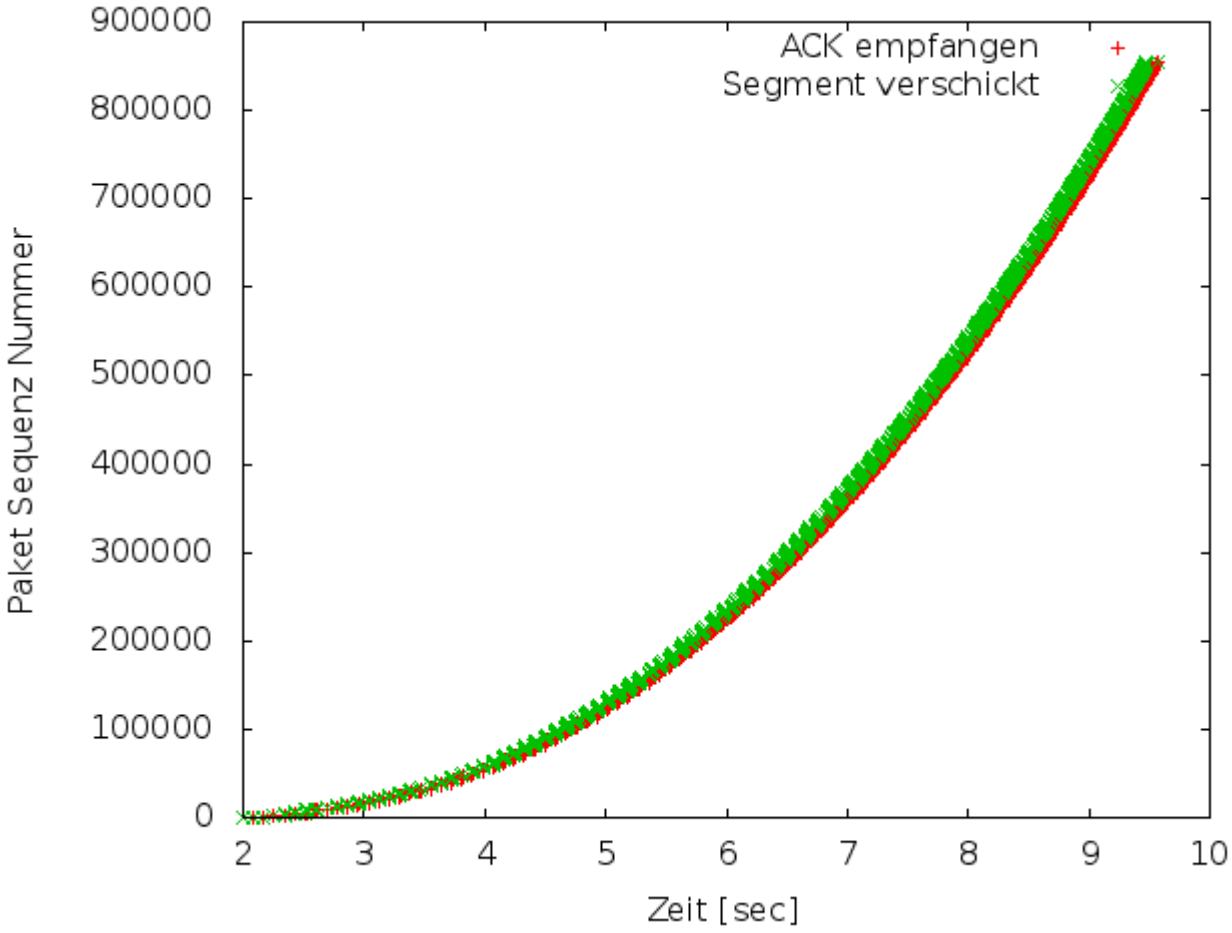

Abbildung 15: TCP Reno komplettes Zeitsequenzdiagramm [Paketverlust: 11]

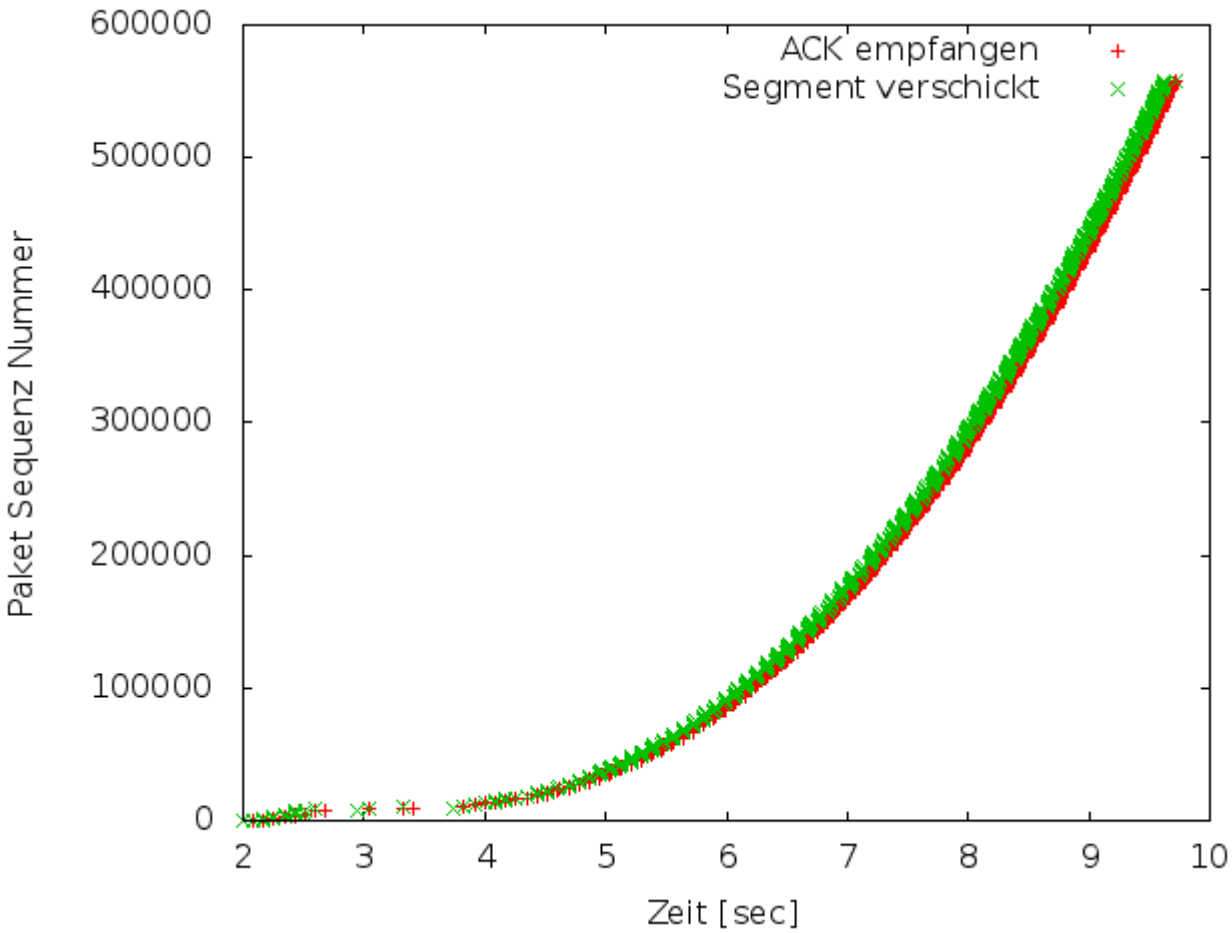

Abbildung 16: TCP Reno komplettes Zeitsequenzdiagramm [Paketverlust: 11, 16, 22]

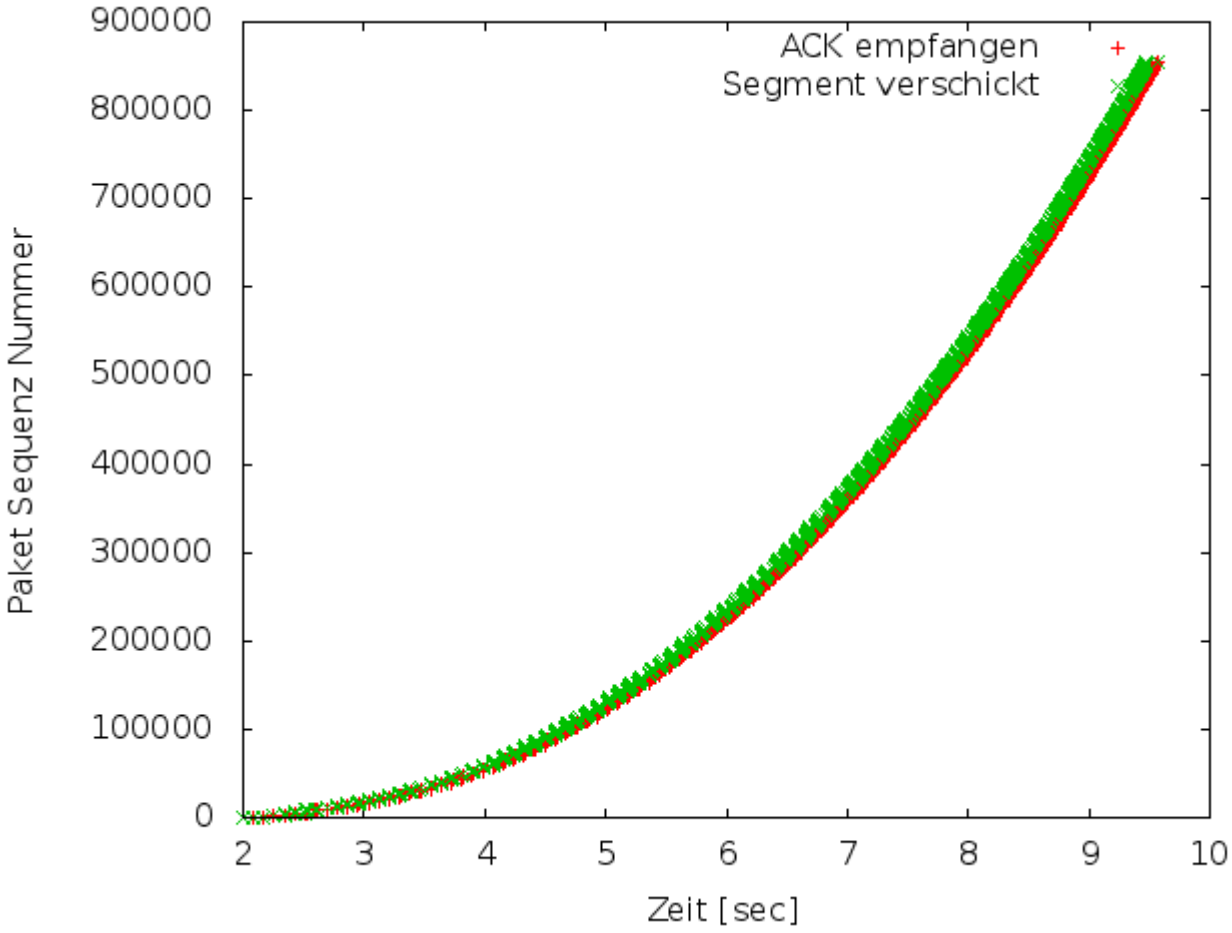

Abbildung 17: TCP New Reno komplettes Zeitsequenzdiagramm [Paketverlust: 11]

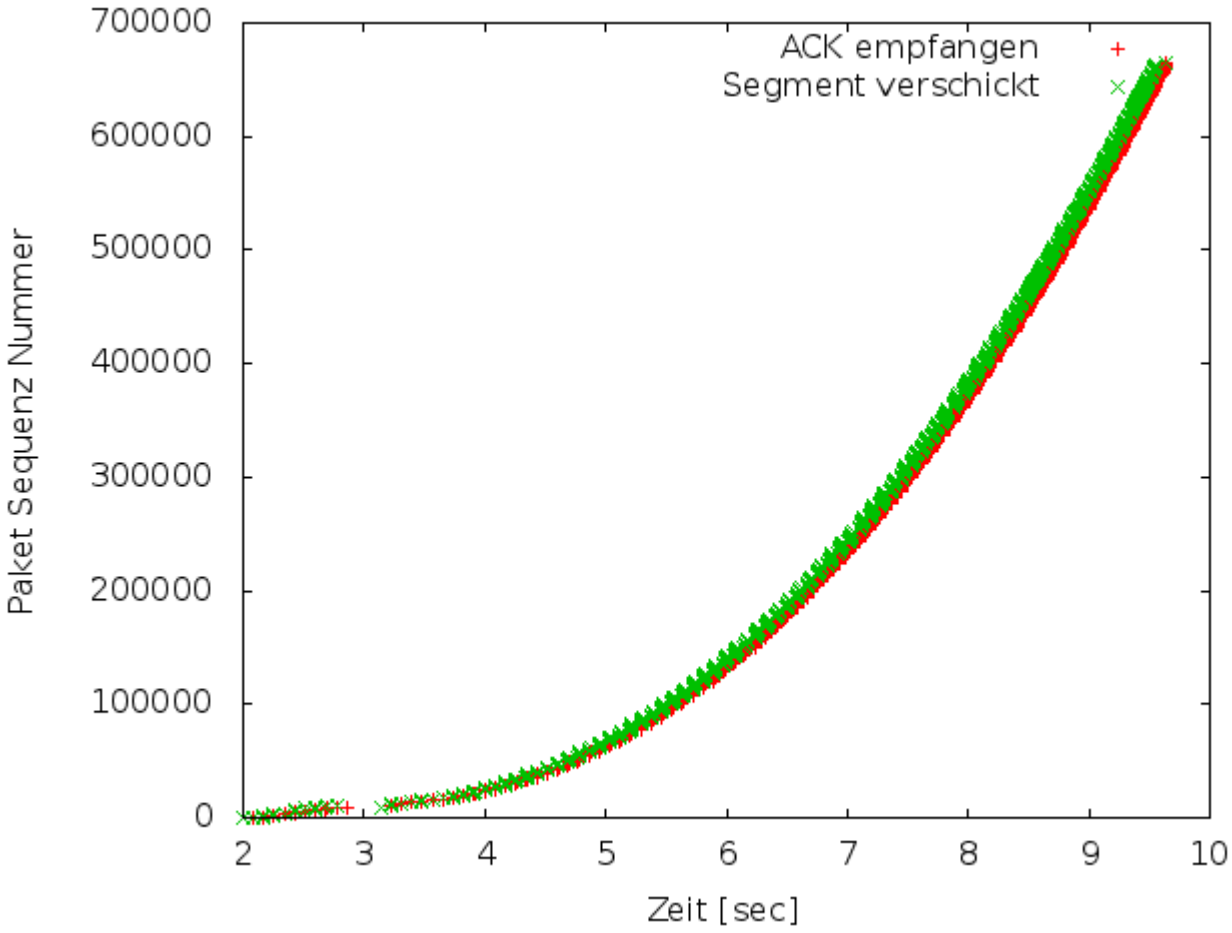

Abbildung 18: TCP New Reno komplettes Zeitsequenzdiagramm [Paketverlust: 11, 16, 22]