Quelle: Computernetzwerke: Der Top-Down-Ansatz, 6. Ausgabe
Jim Kurose, Keith Ross; Pearson, März 2014

Technische Grundlagen der Informatik 2 – Teil 4: Layer 4 TCP

Philipp Rettberg / Sebastian Harnau

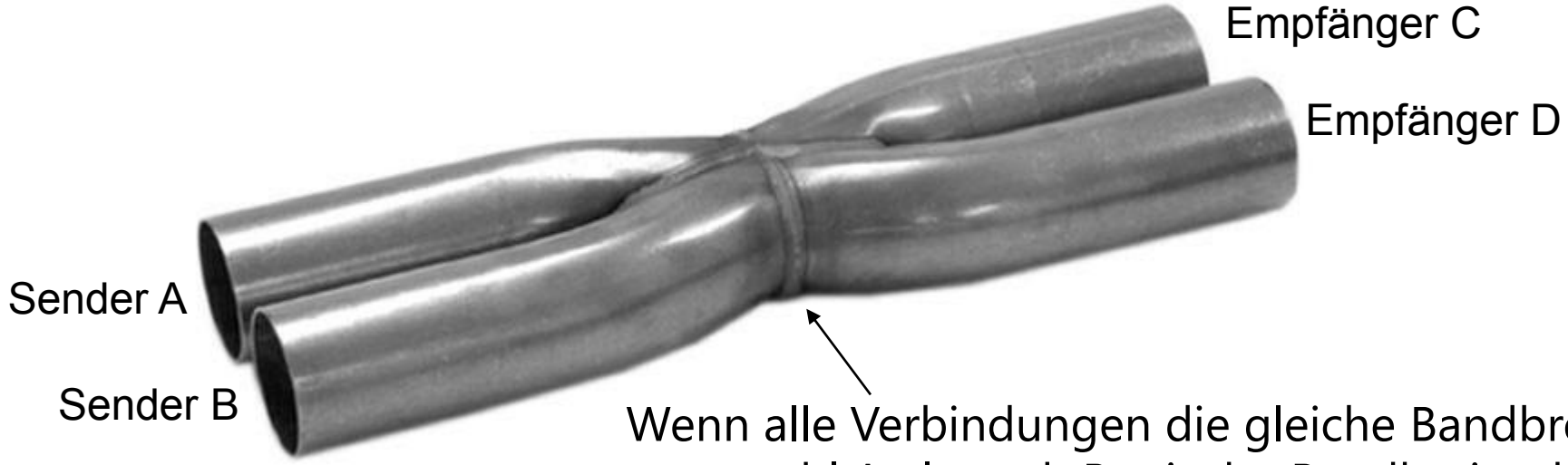
Block 8/18

Transportschicht (Layer 4)

Überlastkontrolle

Grundlagen der Überlastkontrolle

Problem:
Wenn alle Sender im Netz immer so viele Pakete losschicken, wie bei den Empfängern in den Puffer passen, dann kann es zu Überlast (Congestion) im Netz kommen, da nicht auf die momentane Situation im Netz Rücksicht genommen wird.



Wenn alle Verbindungen die gleiche Bandbreite haben und sowohl A als auch B mit der Bandbreite der Verbindung senden, dann kommt es hier zu Netzwerküberlast (Congestion)!

Grundlagen der Überlastkontrolle

- Informell: „Zu viele Systeme senden zu viele Daten, das Netzwerk kann nicht mehr alles transportieren“
- Verschieden von Flusskontrolle (Überlast eines einzelnen Empfängers)!
- Erkennungsmerkmale:
  - Paketverluste (Pufferüberlauf in den Routern)
  - Lange Verzögerungen (Warten in den Routern)

Eines der zentralen Probleme in Computernetzwerken!

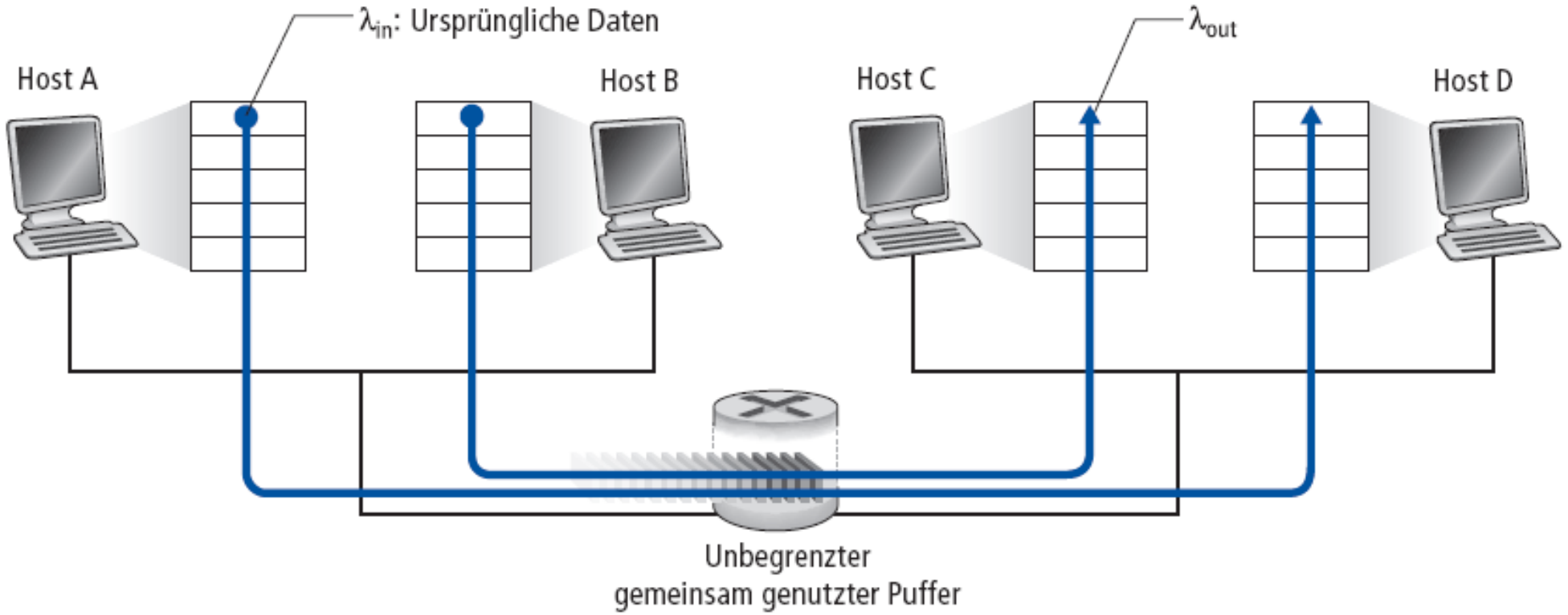
Ursachen und Kosten von Überlast (Szenario 1)

Szenario 1:

Zwei Sender, zwei Empfänger

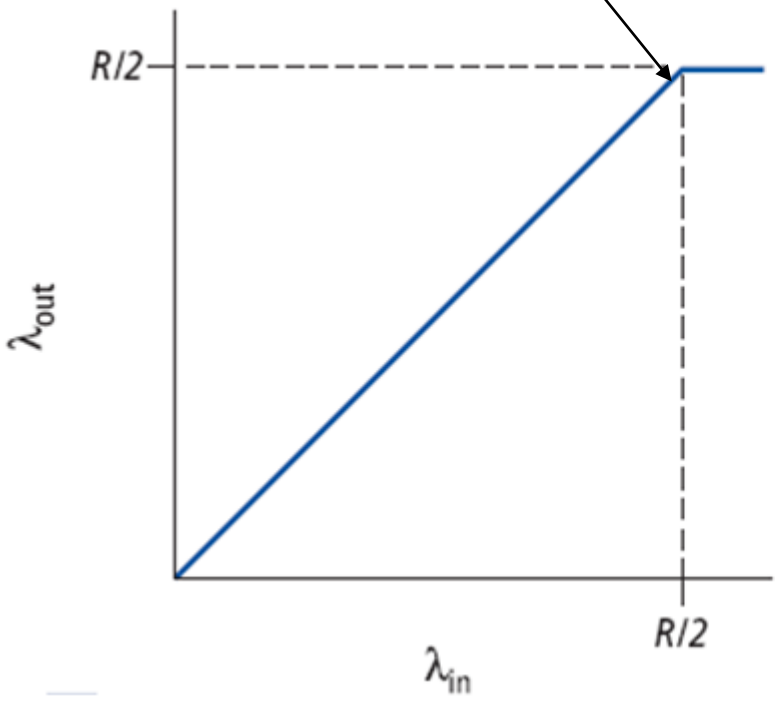
Ein Router, unendlicher Pufferplatz

Keine Übertragungswiederholungen

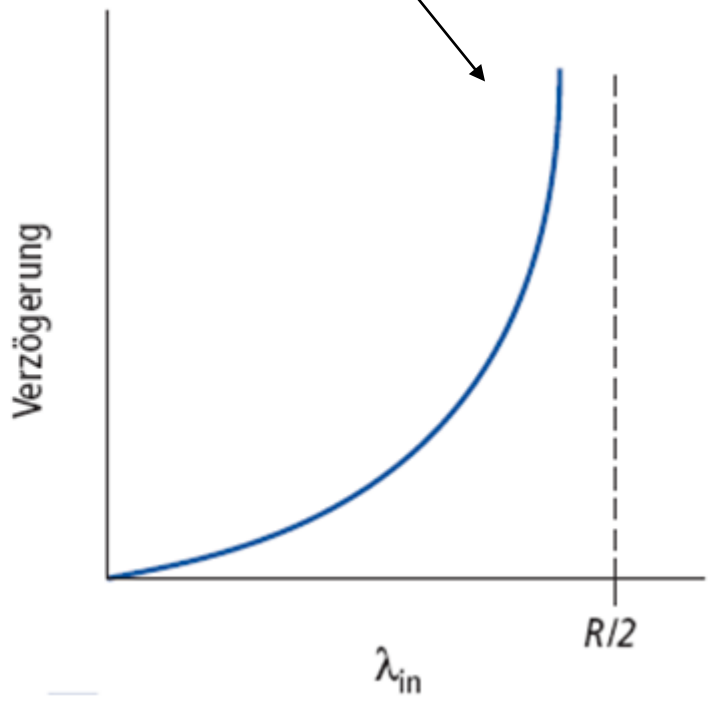


Ursachen und Kosten von Überlast (Szenario 1)

Weitere Erhöhung der Datenmenge pro Zeit beim Sender führt zu keiner Erhöhung der Empfangsmenge pro Zeit-> Stau



Je mehr gesendet wird, desto öfter kommen die Pakete der zwei Sender überschneidend an und es muss gepuffert werden.

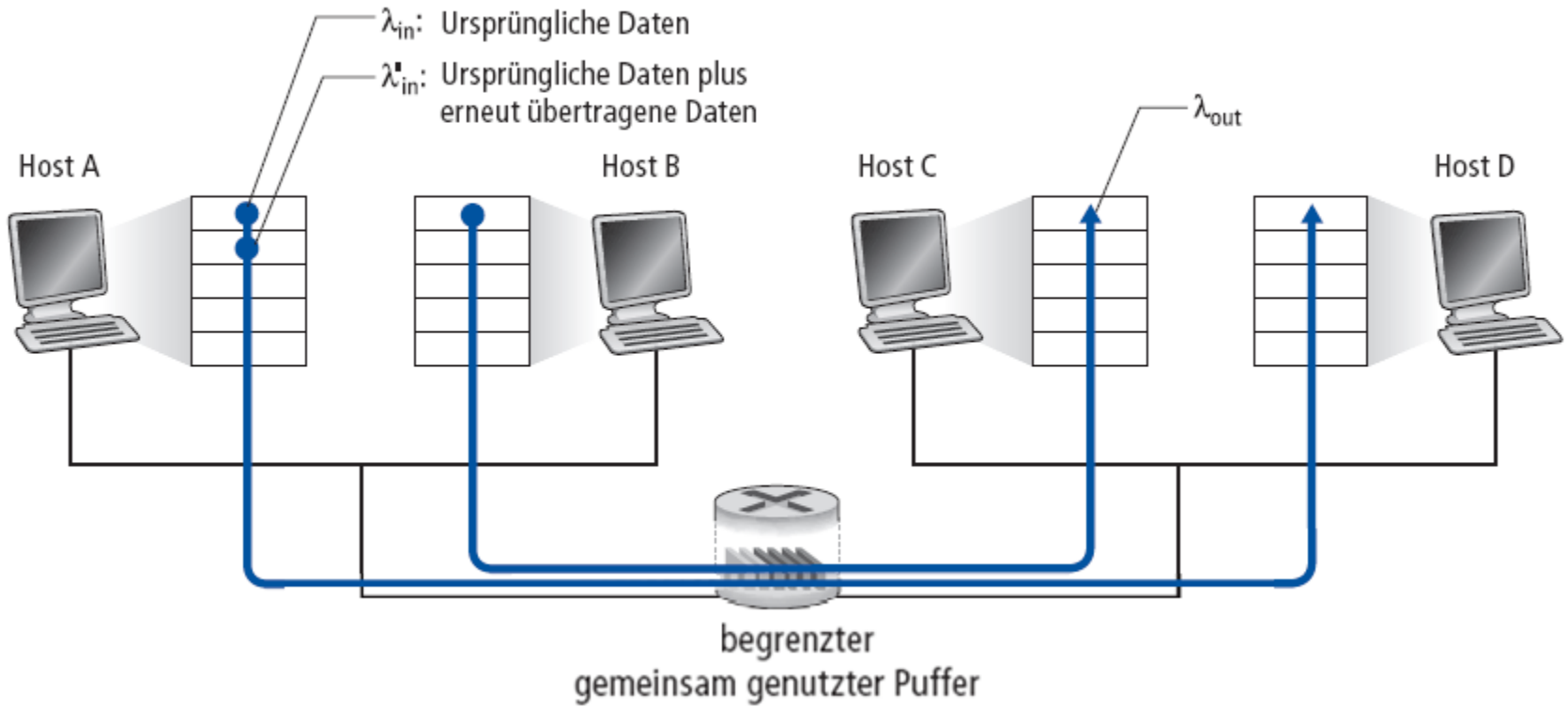


- R : Maximale Kapazität in Byte/Sek
- R/2 : Genutzte Kapazität jedes Senders
- λ_{in} : Gesendete Datenmenge
- λ_{out} : empfangene Datenmenge

Ursachen und Kosten von Überlast (Szenario 2)

Szenario 2:

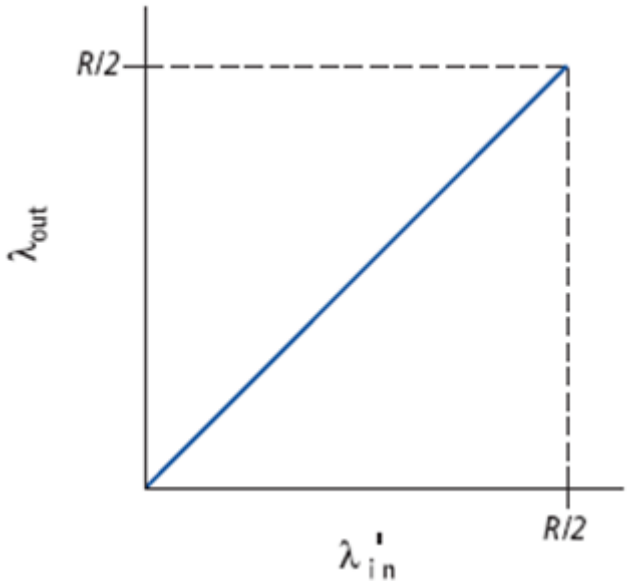
- Ein Router, endlicher Puffer
- Sender wiederholt die Übertragung verlorener Pakete



Ursachen und Kosten von Überlast (Szenario 2)

Ohne Wiederholungen:

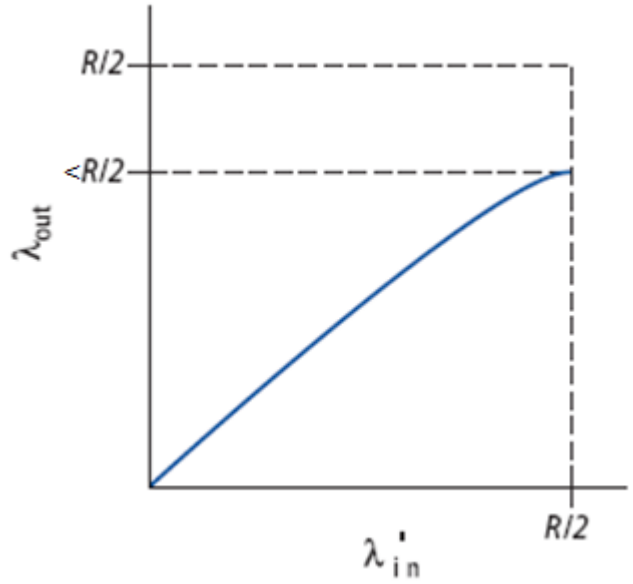
Es gilt: $\lambda'_{in} = \lambda_{in} = \lambda_{out}$



Mit Wiederholungen:

Es gilt: $\lambda_{in} = \lambda_{out}$ (Goodput)

Aber: $\lambda'_{in} > \lambda_{in}$



Ursachen und Kosten von Überlast (Szenario 2)

Beobachtungen:

- Je höher die Auslastung eines Knoten ist, desto exponentiell größer werden die Verzögerungen bei der Datenübertragung

- Überflüssige Übertragungswiederholungen: Leitung befördert mehrere Kopien desselben Pakets

"Kosten" der Überlast:

- Mehr Arbeit (Übertragungswiederholungen) bei gleichem "Goodput"

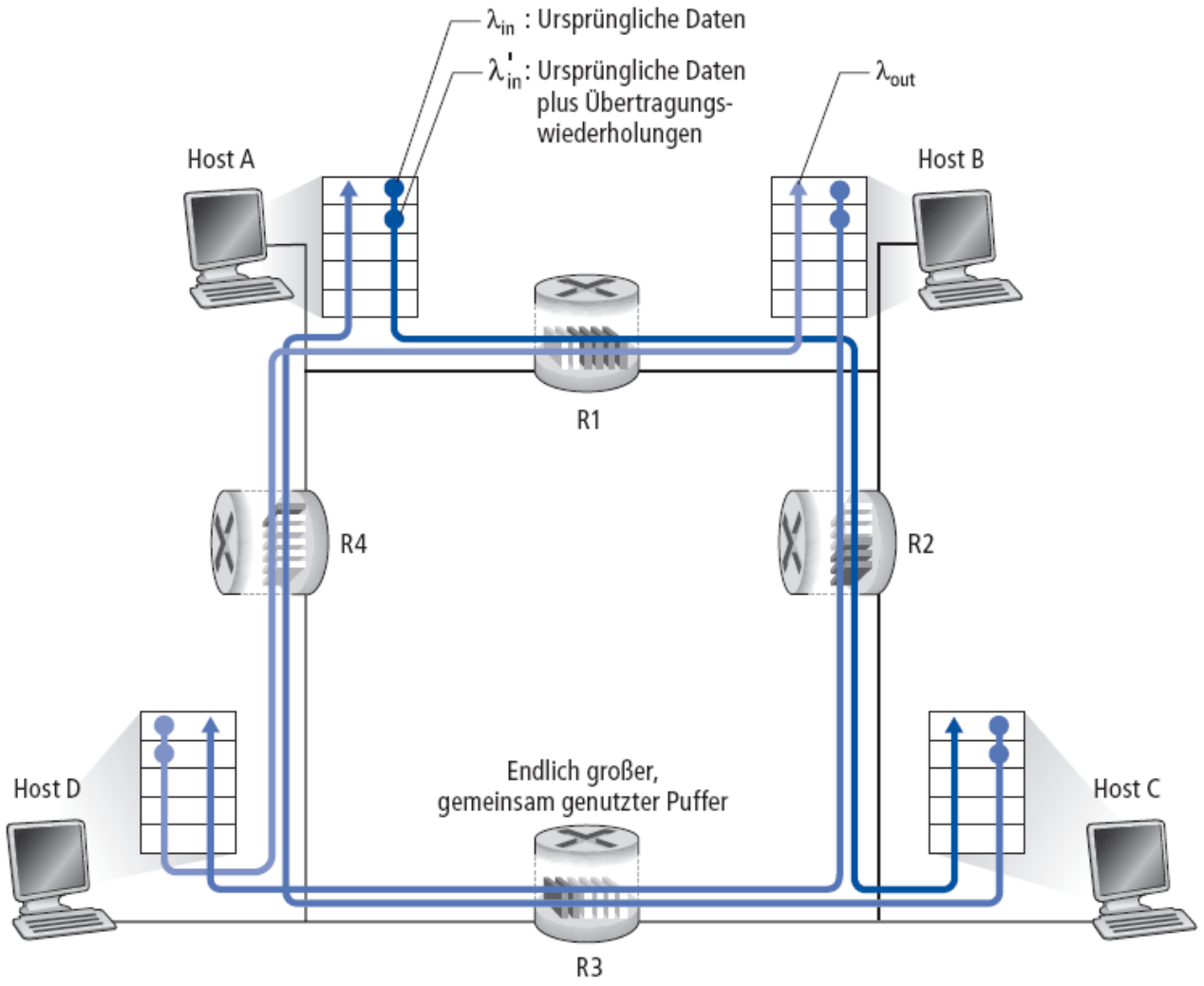
- Maximale Kapazität der Leitung wird nicht ausgeschöpft.

Ursachen und Kosten von Überlast (Szenario 3)

Szenario 3:

- Vier Hosts
- Router auf jedem Wegabschnitt
- Mit Übertragungswiederholungen

Was passiert, wenn λ_{in} groß wird?



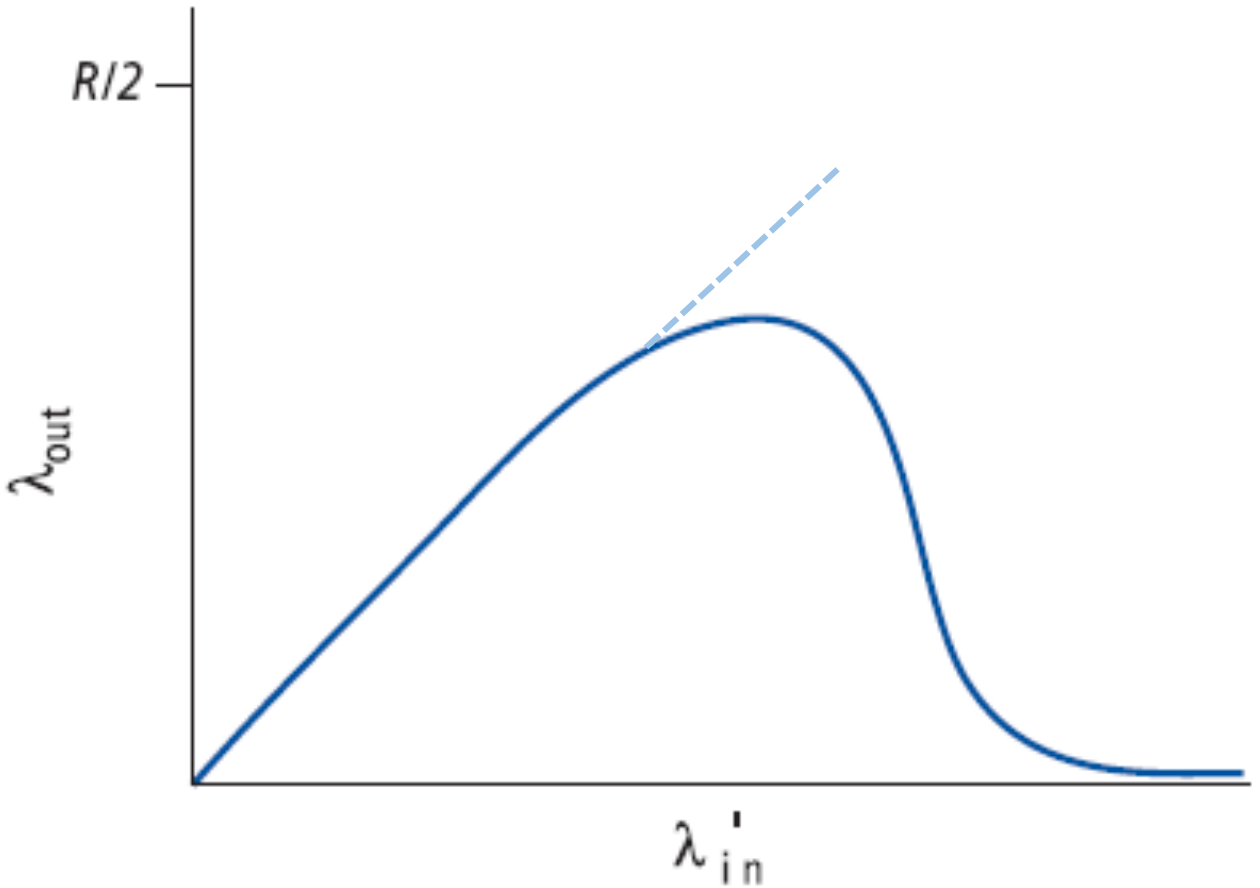
Ursachen und Kosten von Überlast (Szenario 3)

Wenn Pakete verworfen haben, dann haben sie bereits Ressourcen in anderen Routern verbraucht!

Dies kann zu einem **Congestion Collapse** führen:

Die gesamten Ressourcen des Netzwerkes werden für Pakete verbraucht, die nicht an ihr Ziel gelangen!

Die Kommunikation bricht fast komplett zusammen!



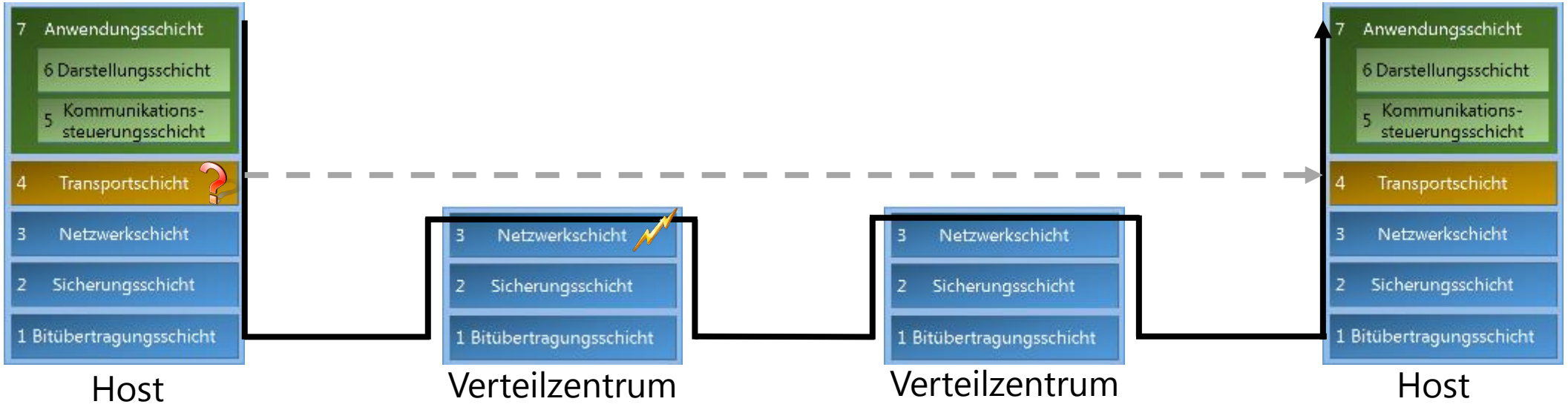
Ursachen und Kosten von Überlast (Szenario 3)

Beobachtungen:

- Der Weg eines Pakets durch das Netz ist nicht vorbestimmbar.
- Zu viel Last kann das Netz kollabieren lassen.

Fragen:

- Wie kann der Sender merken, dass er zu viel sendet?
- Beachten Sie: Wir sind auf Layer 4. Die Überlast tritt auf Layer 3 auf…



Ansätze zur Kontrolle von Überlast

Netzwerkunterstützt:

Router geben den Endsystemen Hinweise:

- Durch ein einzelnes Bit, welches im Paketheader von den Routern gesetzt werden kann
- Durch Vorgabe einer expliziten Senderate
- ➢ Verletzt das Prinzip der Unabhängigkeit der Schichten

Ende-zu-Ende (TCP):

- Keine explizite Unterstützung durch das Netzwerk
- Überlast wird von den Endsystemen durch Paketverlust und erhöhte Verzögerung festgestellt

TCP-Überlastkontrolle: Einführung

- Wie erkennt der Sender Überlast?
  - Verlustereignis = Timeout oder drei doppelte ACKs
  - TCP: Sender verringert sein Congestion Window und damit seine Rate nach einem Verlustereignis
- Congestion Window (CongWin):
  - Menge der maximal zulässigen unbestätigten Daten
  - CongWin/MSS=Maximale Anzahl gleichzeitig unbestätigter Segmente

TCP-Überlastkontrolle: Einführung

Sender begrenzt die Übertragung: $\text{LastByteSent} - \text{LastByteAcked} \leq \text{CongWin}$

Allgemein gilt:

$$\text{Rate} = \frac{\text{CongWin}}{\text{RTT}} \quad \text{Byte/s}$$

CongWin ist dynamisch, hängt von der wahrgenommenen Netzwerklast ab

Drei Mechanismen notwendig:

- Slow Start
- Additive Increase, Multiplicative Decrease (AIMD)
- Vorsichtiges Verhalten nach einem Timeout

TCP-Überlastkontrolle: Slow Start

Bei Verbindungsbeginn: **CongWin** = 1 MSS (Maximum Segment Size)

- Beispiel:
  - MSS = 500 Byte
  - RTT = 200 ms
  - Initiale Rate = 500 / 0,2 * 8=20 kBit/s

$$\text{Rate} = \frac{\text{CongWin}}{\text{RTT}} \quad \text{Byte/s}$$

- Verfügbare Bandbreite kann viel größer als MSS/RTT sein.
- Die Rate sollte sich schnell der verfügbaren Rate anpassen.

➢ Exponentielle Erhöhung pro RTT, bis es zum ersten Verlustereignis kommt.

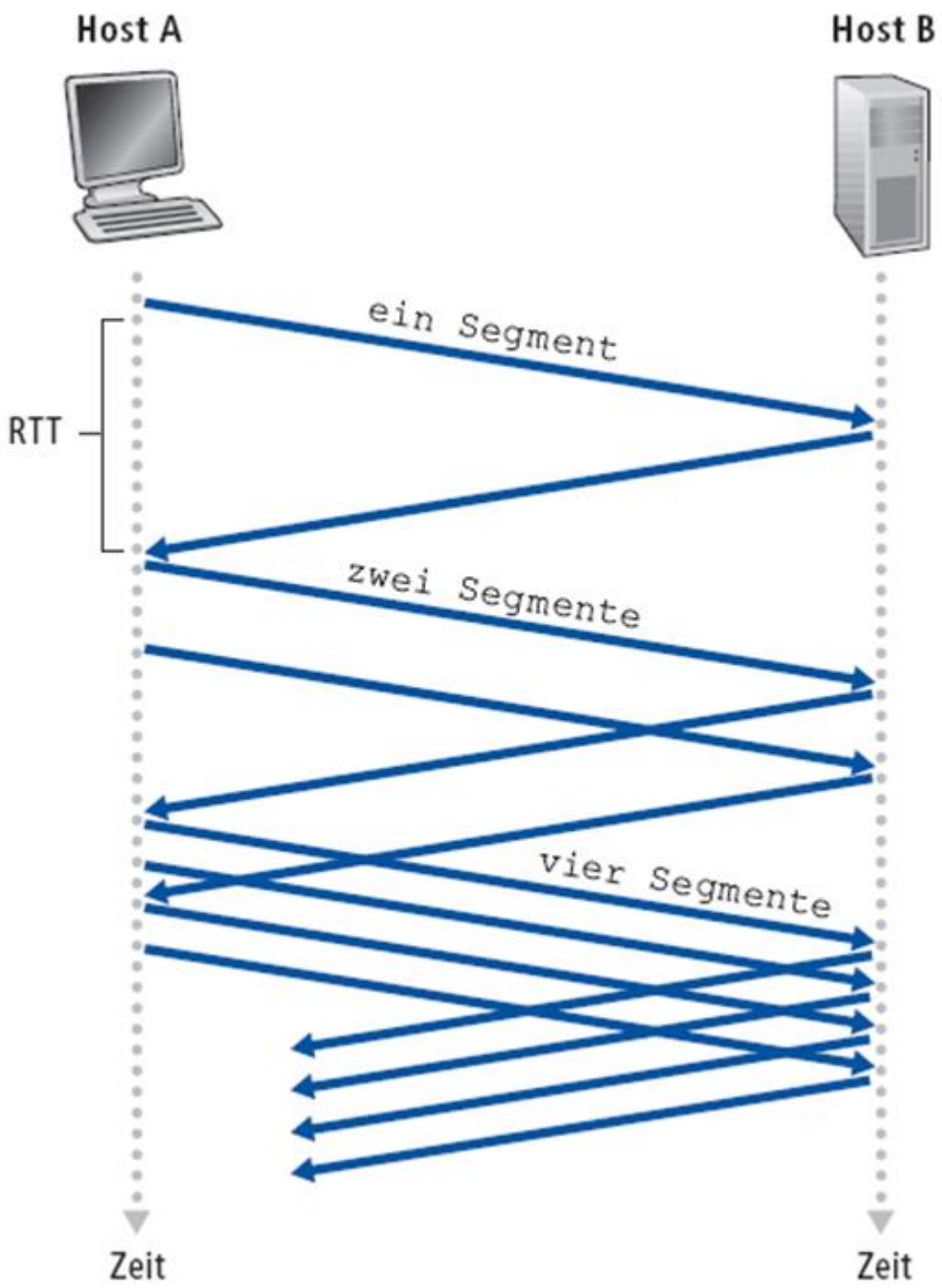
TCP-Überlastkontrolle: Slow Start

Bei Verbindungsbeginn: Erhöhe die Rate exponentiell schnell bis zum ersten Verlust-ereignis

- Verdoppeln von **CongWin** in jeder RTT
- Realisiert: **CongWin** um 1 für jedes erhaltene ACK erhöhen

Ergebnis:

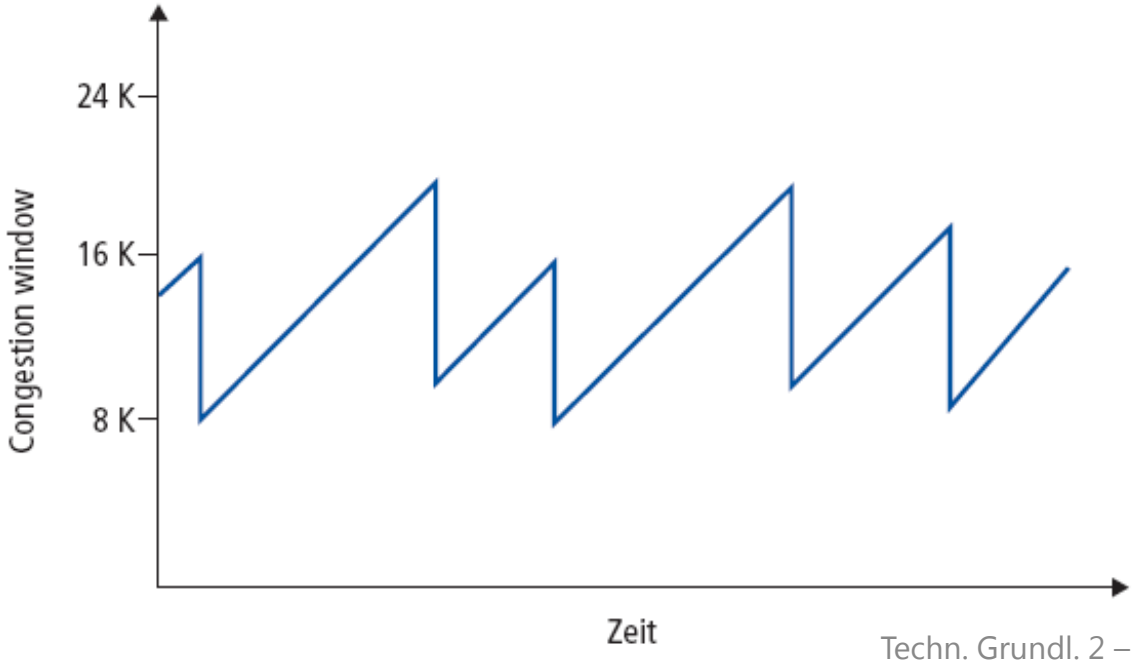
Initiale Rate ist gering, wächst aber exponentiell schnell



TCP-Überlastkontrolle: Additive Increase, Multiplicative Decrease (AIMD)

Ansatz bei Erreichung eines vorher definieren Schwellwertes: Erhöhe die Übertragungsrate (Fenstergröße=CongWin), um nach überschüssiger Bandbreite zu suchen, bis ein Verlust eintritt.

- **Additive Increase**: Erhöhe CongWin um eine MSS (Maximum Segment Size) pro RTT, bis ein Verlust erkannt wird.
- **Multiplicative Decrease**: Halbiere CongWin, wenn ein Verlust erkannt wird.



Sägezahnverlauf:
nach überschüssiger Bandbreite suchen

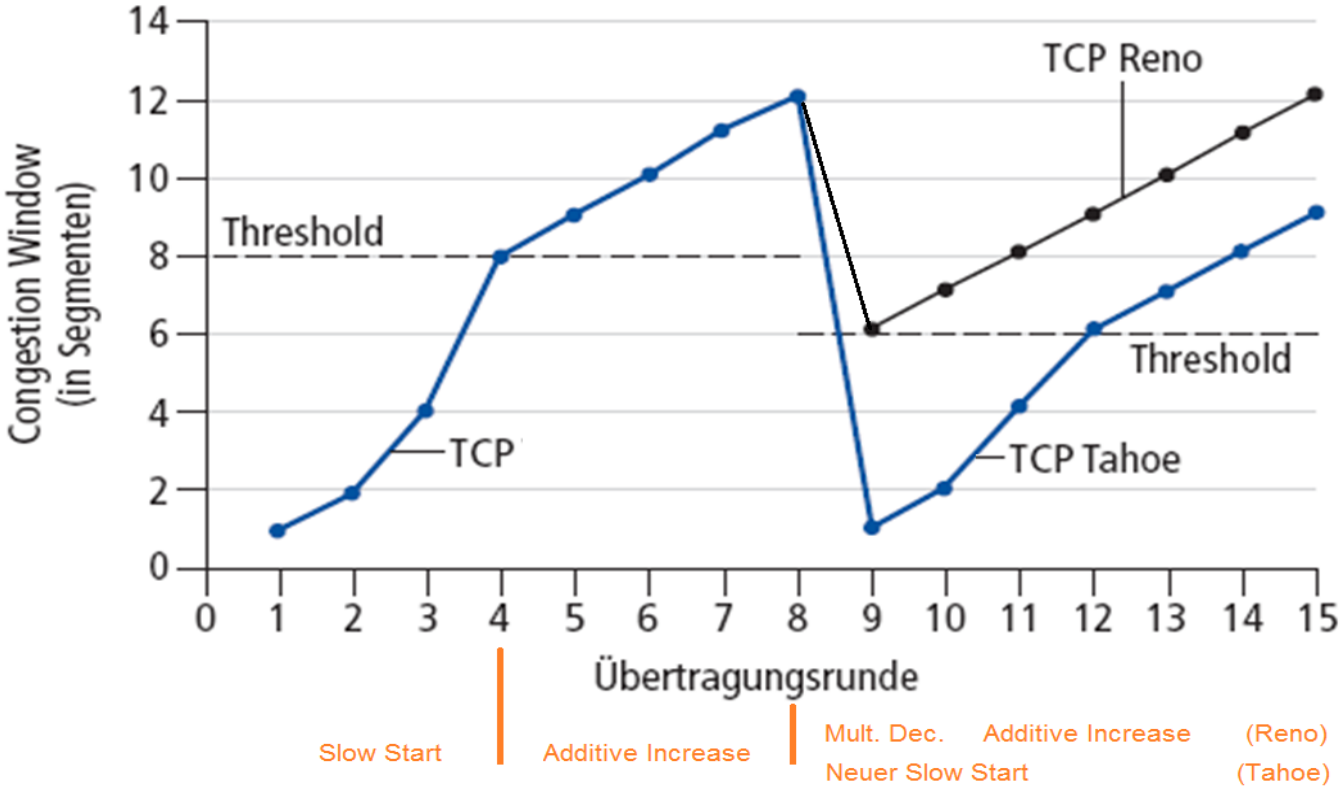
TCP-Überlastkontrolle: AIMD-Umsetzung: Congestion Avoidance

Frage: Wann gehen wir vom exponentiellen Wachstum zum linearen Wachstum über?

Antwort: Wenn **CongWin** die Hälfte des Wertes vor dem letzten Verlustereignis erreicht hat

Implementierung:

- Variabler Threshold
- Bei einem Verlustereignis wird Threshold auf die Hälfte von CongWin vor dem Verlustereignis gesetzt
- Neuer Wert für CongWin:
  - TCP Tahoe (alt): 1
  - TCP Reno (neu): Treshold



Details: Verluste erkennen bei TCP Reno

Nach drei doppelten ACKs:

- CongWin halbieren, Threshold auf diesen Wert setzen
- Fenster wächst dann linear

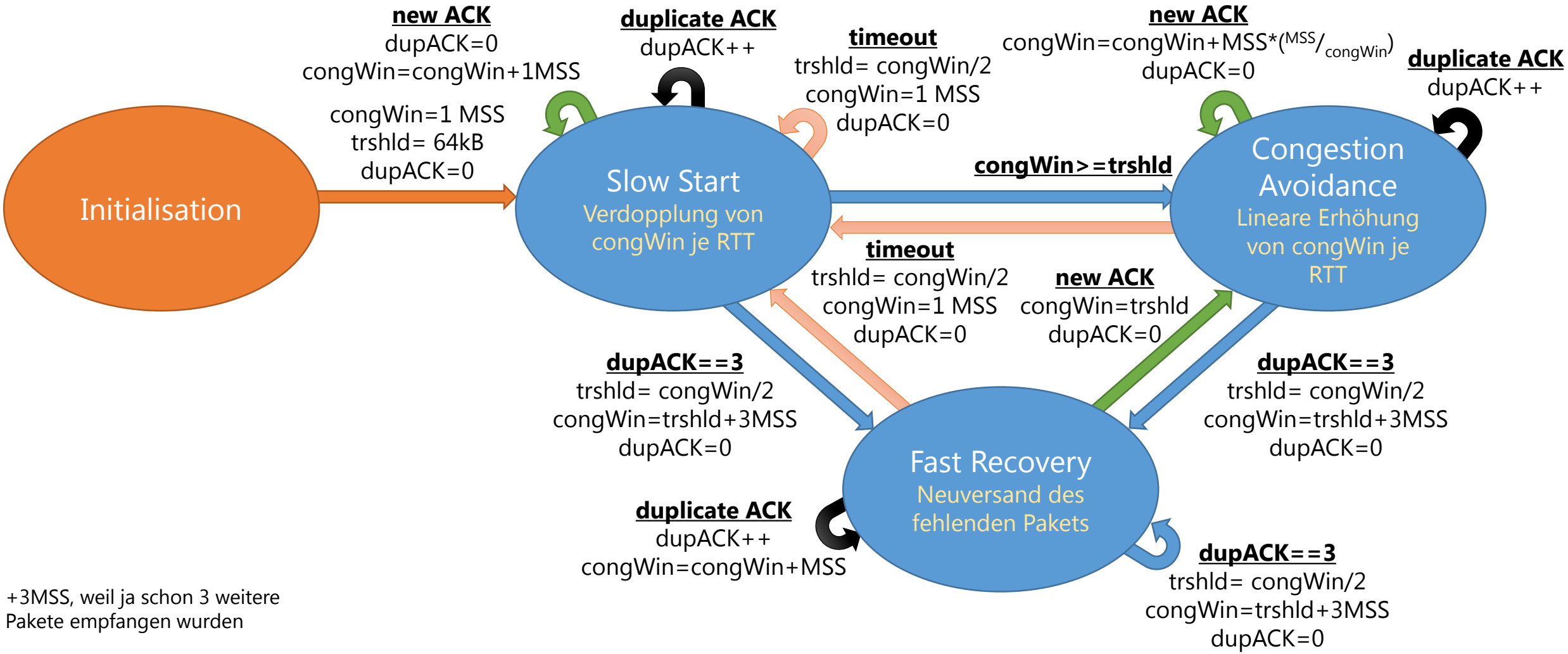
Nach Timeout:

- CongWin auf eine MSS setzen, Threshold auf die Hälfte des alten CongWin setzen
- Fenster wächst dann exponentiell, bis Threshold erreicht ist (Slow Start)
- … danach linear

Philosophie:

- Drei doppelte ACKs zeigen an, dass das Netzwerk in der Lage ist, Pakete auszuliefern
- Timeout ist "schlimmer", weil gar keine Rückmeldung mehr erfolgt

Phasen der TCP-Überlastkontrolle



+3MSS, weil ja schon 3 weitere Pakete empfangen wurden

Zusammenfassung: TCP-Überlastkontrolle

Wenn CongWin kleiner als Threshold ist, befindet sich der Sender in der Slow-Start-Phase, das Fenster wächst exponentiell.

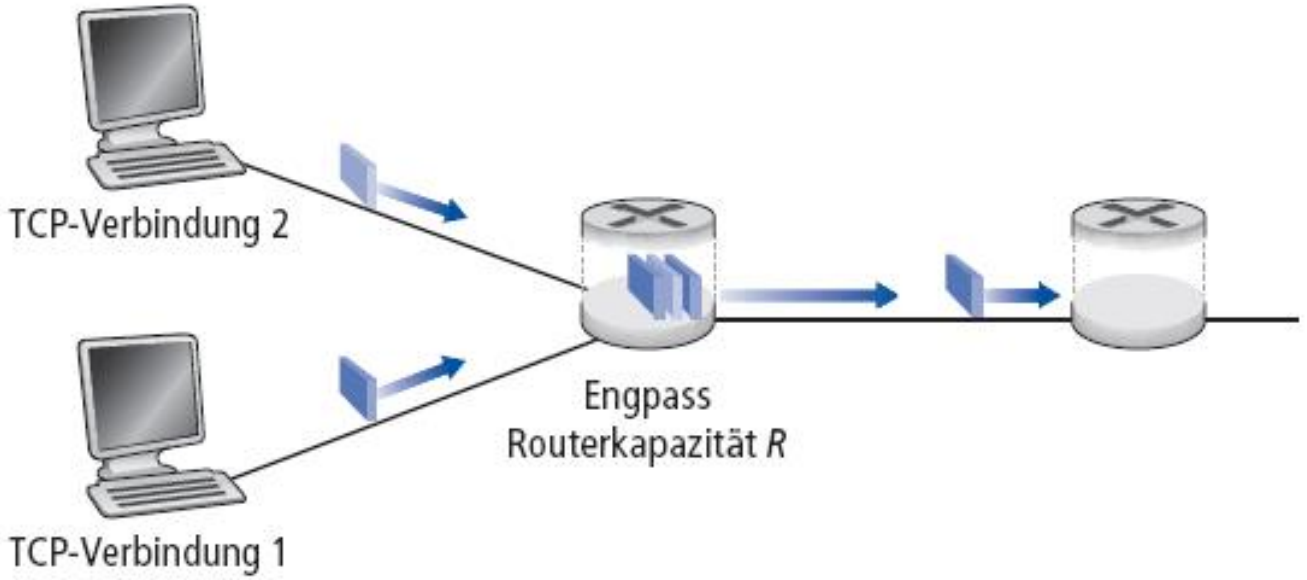
Wenn CongWin größer als Threshold ist, befindet sich der Sender in der Congestion-Avoidance-Phase, das Fenster wächst linear.

Wenn ein Triple Duplicate ACK auftritt (drei doppelte ACKs für dasselbe Segment), wird Threshold auf CongWin/2 gesetzt und dann CongWin auf Threshold .

Wenn ein Timeout auftritt, werden Threshold auf CongWin/2 und CongWin auf eine 1 MSS gesetzt.

TCP-Fairness

Faires Ziel: Wenn k TCP-Sitzungen sich denselben Engpass mit Bandbreite R teilen, dann sollte jede eine durchschnittliche Rate von circa R/k erhalten.

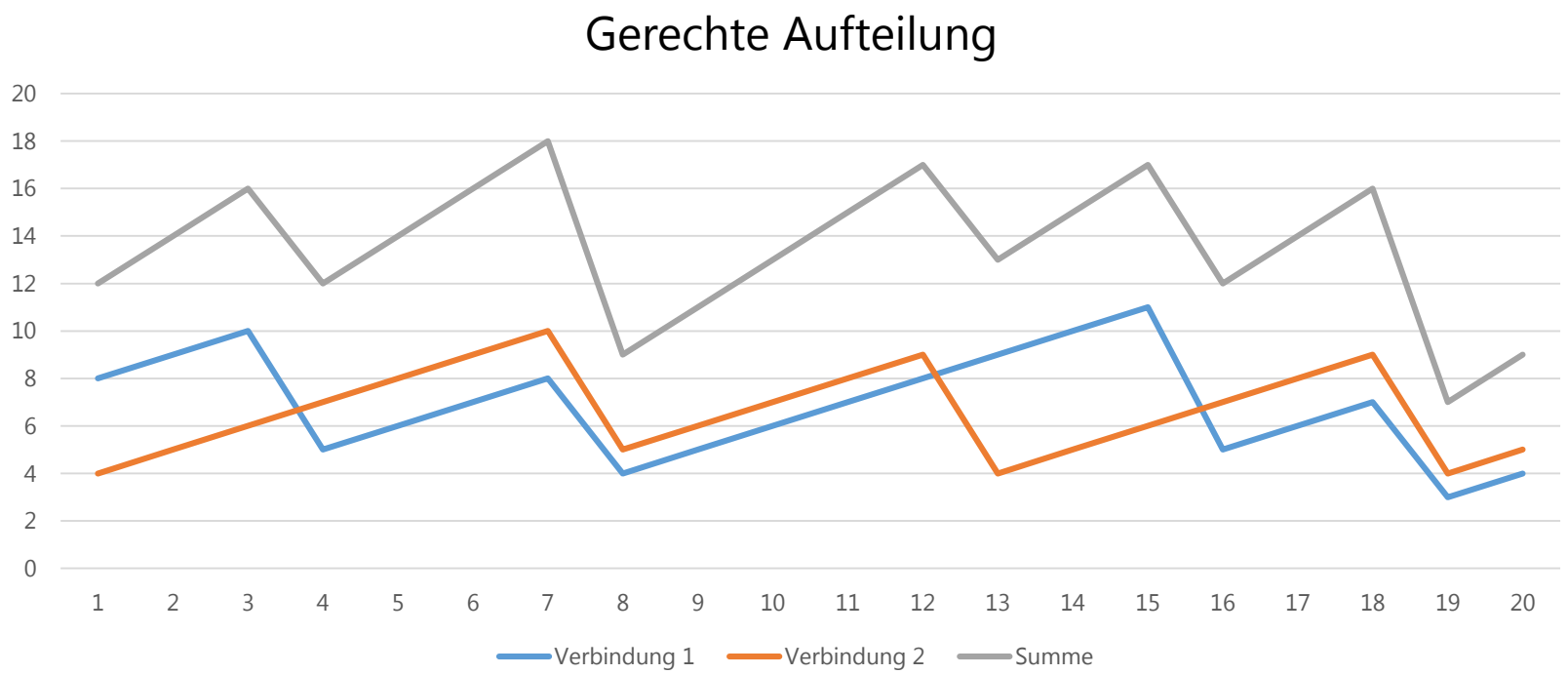


Warum ist TCP fair?

Zwei Verbindungen im Wettbewerb:

Additive Increase führt zu einer Steigung von 1, wenn der Durchsatz wächst

Multiplicative Decrease reduziert den Durchsatz proportional



Mehr zum Thema Fairness

Fairness und UDP

Viele Multimedia-Anwendungen verwenden kein TCP

- Sie wollen nicht, dass die Rate durch Überlastkontrolle reduziert wird

Stattdessen: Einsatz von UDP

- Audio-/Videodaten mit konstanter Rate ins Netz leiten, Verlust hinnehmen

Forschungsgebiet: TCP- Friendliness

Fairness und parallele TCP-Verbindungen:

- Eine Anwendung kann zwei oder mehr parallele TCP-Verbindungen öffnen
- Webbrowser machen dies häufig
- Beispiel: Engpass hat eine Rate von R, bisher existieren neun Verbindungen
  - Neue Anwendung legt eine neue TCP-Verbindung an und erhält die Rate R/10
  - Neue Anwendung legt elf neue TCP-Verbindungen an und erhält mehr als R/2!

Ausblick

- Es gibt Situationen, in denen kein Protokoll ideal funktioniert.
- Die Entwicklung eines fairen Protokolls mit Überlastkontrolle und deutlich gedämpften AIMD-Sägezahn ist ein wichtiges Forschungsgebiet.
- Ob sich die neuen Protokolle erfolgreich verbreiten werden, muss die Zukunft zeigen.
- Neue Protokolle sind „besser“, aber TCP und UDP sind „gut genug“.
- Wird sich „besser“ durchsetzen? Ist „gut genug“ irgendwann nicht mehr ausreichend?

Ausblick

SCTP (Stream Control Transmission Protocol – RFC 4960):

- Multiplexen mehrerer verschiedener Ströme (Multidatenstrom). Verlust in einem Strom beeinflusst die anderen nicht! Datenzustellung außerhalb der Reihenfolge erlaubt.

DCCP (Datagram Congestion Control Protocol – RFC 4340):

- UDP-ähnlich, aber mit TCP-kompatibler Überlastkontrolle. Kompromiss zwischen zeitnaher und zuverlässiger Zustellung mit Reaktion auf Überlast.

TFRC (TCP Friendly Rate Control Protocol – RFC 5348):

- Eher ein Überlastkontrollmechanismus denn den ein vollständiges Protokoll. Ziel: Glättung der Sägezahnstruktur.