Stauvermeidung in TCP Tahoe

CoNe Freiburg

x: Anzahl Pakete pro RTT

- Jacobson (1988):
  - Parameter: cwnd und Slow-Start-Schwellwert (ssthresh=slow start threshold)
  - S = Datensegmentgröße = maximale Segmentgröße (MSS)
- Verbindungsaufbau:
  - cwnd ← S ssthresh ← 65535

x ← 1 **y ← max**

- Bei Paketverlust, d.h. Bestätigungsdauer > RTO,
  - multiplicatively decreasing

  cwnd ← S ssthresh ← $\max\left\{2 \times S, \frac{\min\{cwnd, wnd\}}{2}\right\}$

x ← 1 **y ← x/2**

- Werden Segmente bestätigt und cwnd ≤ ssthresh, dann
  - slow start: cwnd ← cwnd + S

x ← 2·x, bis x = y

- Werden Segmente bestätigt und cwnd > ssthresh, dann additively increasing

$$\text{cwnd} \leftarrow \text{cwnd} + S \frac{S}{cwnd}$$

x ← x +1

UNI FREIBURG

TCP Tahoe

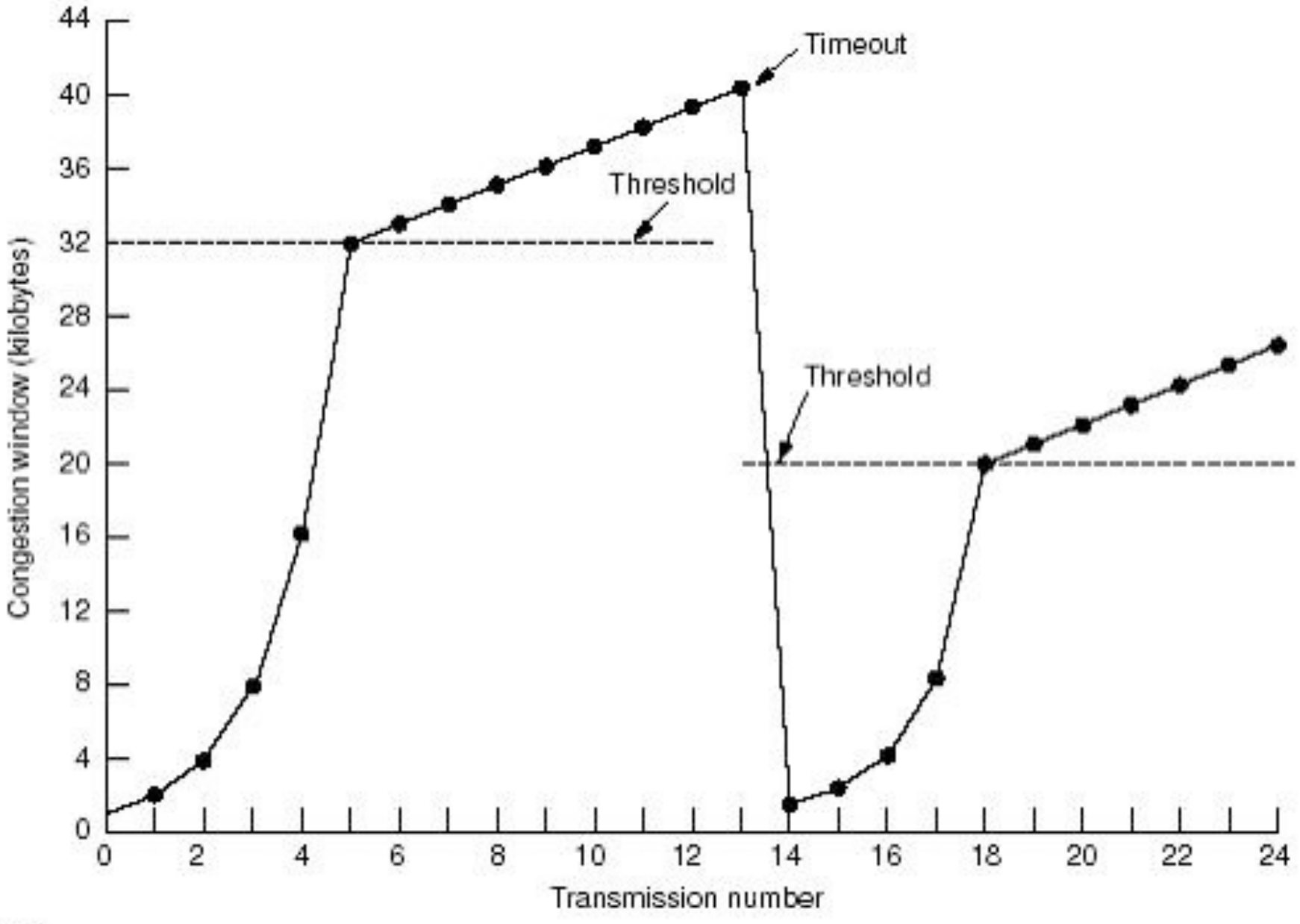
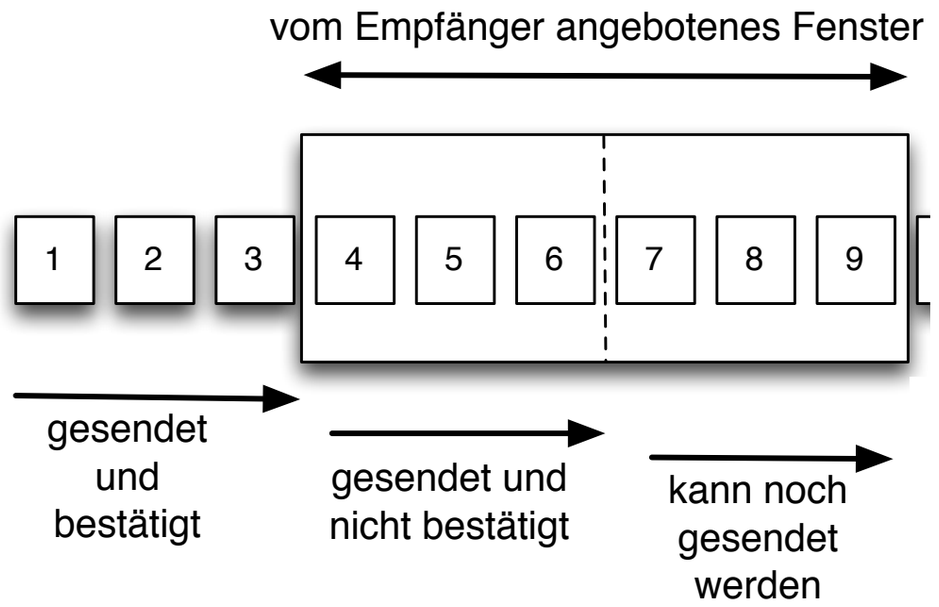


Fig3

pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

Fast Retransmit und Fast Recovery

- TCP Tahoe [Jacobson 1988]:
  - Geht nur ein Paket verloren, dann
    - Wiederversand Paket + Restfenster
    - Und gleichzeitig Slow Start
  - **Fast retransmit**: Nach vier Bestätigungen desselben Pakets (“triple duplicate ACK”), sende Paket nochmal, starte mit Slow Start



- TCP Reno [Stevens 1994]
  - Nach Fast retransmit:
    - ssthresh ← min(wnd,cwnd)/2
    - cwnd ← ssthresh + 3 S

y ← x/2

x ← y + 3

  - **Fast recovery** nach Fast retransmit
    - Erhöhe Paketrate mit jeder weiteren Bestätigung
    - cwnd ← cwnd + S
  - Congestion avoidance:
    Trifft Bestätigung von P+x ein: cwnd ← ssthresh

Stauvermeidungsprinzip: AIMD

- Kombination von TCP und Fast Recovery verhält sich im wesentlichen wie folgt:
  - Verbindungsaufbau:

$$x \leftarrow 1$$

  - Bei Paketverlust, MD:multiplicative decreasing

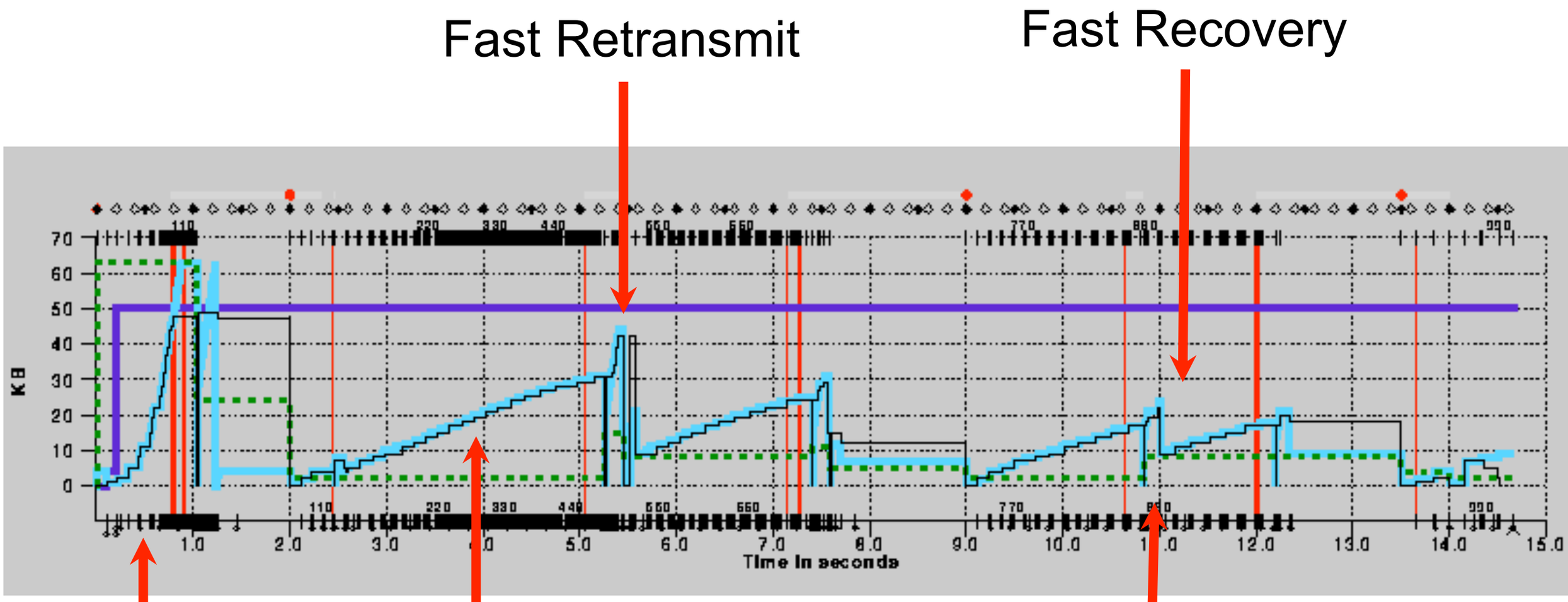
$$x \leftarrow x/2$$

  - Werden Segmente bestätigt, AI: additive increasing

$$x \leftarrow x + 1$$

CoNe Freiburg

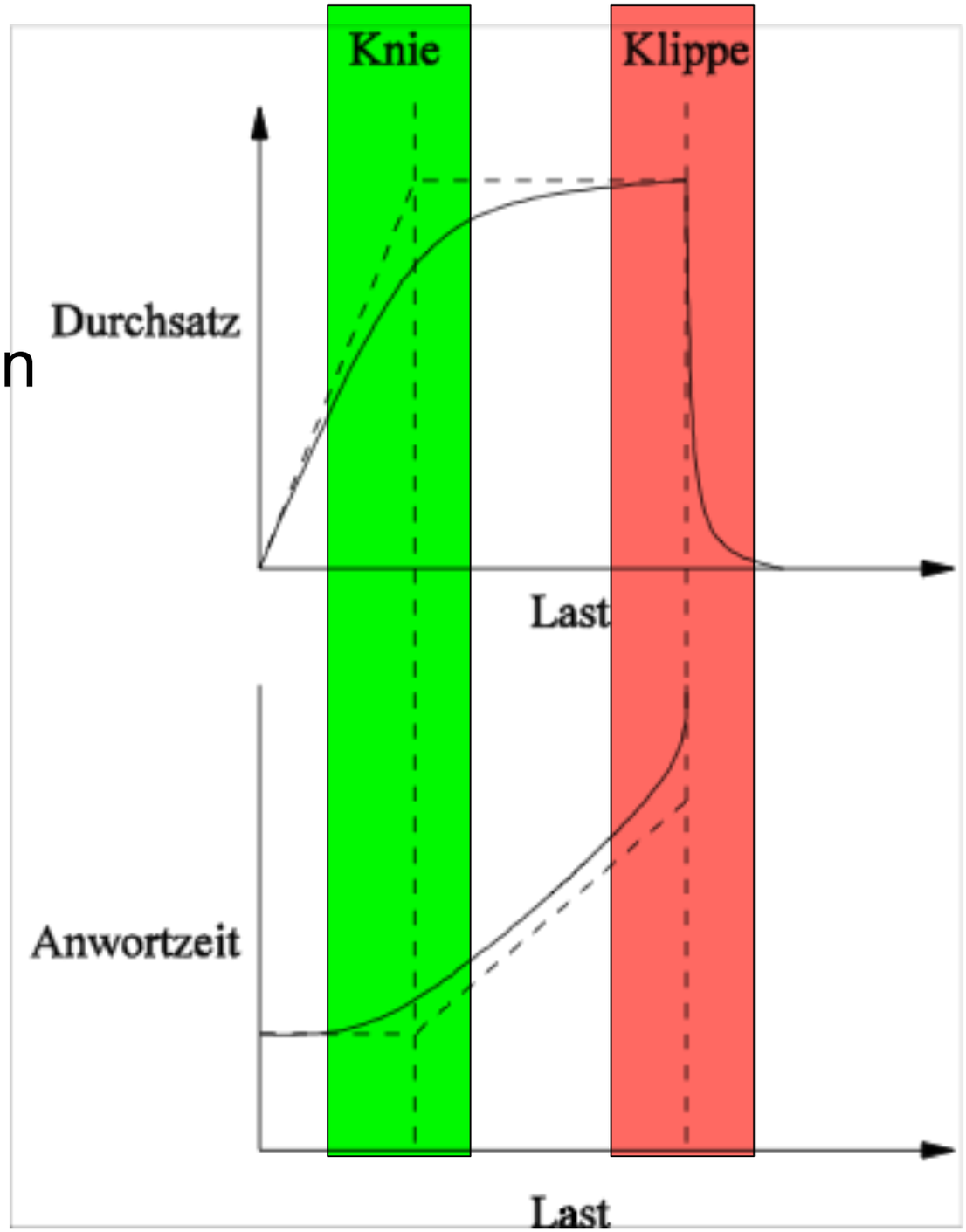
Beispiel: TCP Reno in Aktion



CoNe Freiburg

Durchsatz und Antwortzeit

- Klippe:
  - Hohe Last
  - Geringer Durchsatz
  - Praktisch alle Daten gehen verloren
- Knie:
  - Hohe Last
  - Hoher Durchsatz
  - Einzelne Daten gehen verloren



UNI FREIBURG

CoNe Freiburg

Ein einfaches Datenratenmodell

- n Teilnehmer, Rundenmodell
  - Teilnehmer i hat Datenrate $x_i(t)$
  - Anfangsdatenrate $x_1(0), \ldots, x_n(0)$ gegeben
- Feedback nach Runde t:
  - y(t) = 0, falls $\sum_{i=1}^{n} x_i(t) \leq K$
  - y(t) = 1, falls $\sum_{i=1}^{n} x_i(t) > K$
  - wobei K ist Knielast
- Jeder Teilnehmer aktualisiert in Runde t+1:
  - $x_i(t+1) = f(x_i(t),y(t))$
  - Increase-Strategie $f_0(x) = f(x,0)$
  - Decrease-Strategie $f_1(x) = f(x,1)$
- Wir betrachten lineare Funktionen:

$$f_0(x) = a_I + b_I x \qquad \text{und} \qquad f_1(x) = a_D + b_D x \ .$$

CoNe Freiburg

Lineare Datenratenanpassung

- Interessante Spezialfälle:
  - AIAD: Additive Increase Additive Decrease

$$f_0(x) = a_I + x \qquad \text{und} \qquad f_1(x) = a_D + x \ ,$$

wobei $a_I > 0$ und $a_D < 0$.

  - MIMD: Multiplicative Increase/Multiplicative Decrease

$$f_0(x) = b_I x \qquad \text{und} \qquad f_1(x) = b_D x \ ,$$

wobei $b_I > 1$ und $b_D < 1$.

  - AIMD: Additive Increase Multiplicative Decrease

$$f_0(x) = a_I + x \qquad \text{und} \qquad f_1(x) = b_D x \ ,$$

wobei $a_I > 0$ und $b_D < 1$.

UNI FREIBURG

CoNe Freiburg

Fairness und Effizienz

- Effizienz
  - Last: $$X(t) := \sum_{i=1}^{n} x_i(t)$$
  - Maß $$|X(t) - K|$$
- Fairness: Für x=(x1, …, xn):

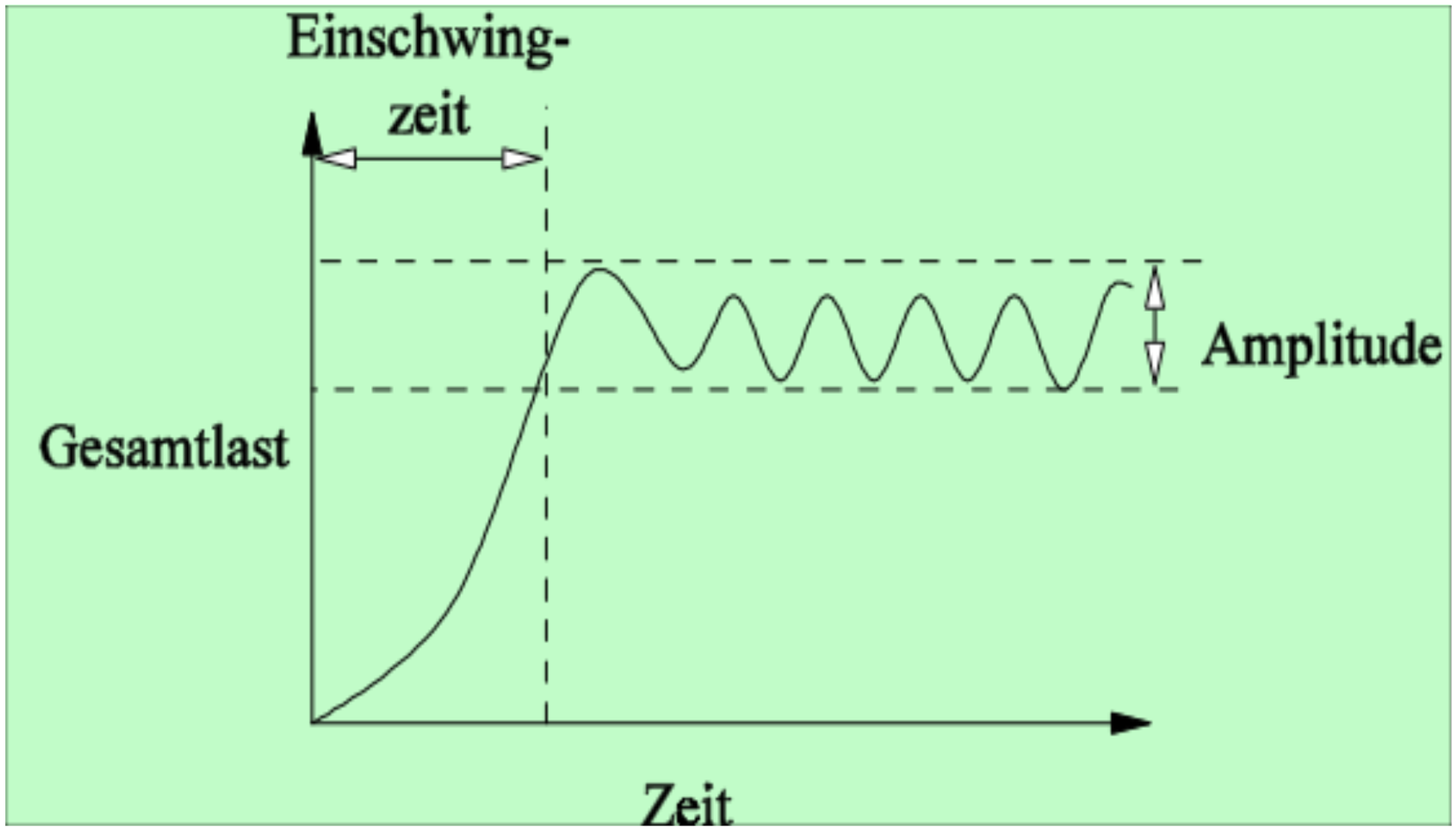
$$F(x) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} (x_i)^2} .$$

- 1/n ≤ F(x) ≤ 1
- F(x) = 1 ↔ absolute Fairness
- Skalierungsunabhängig
- Kontinuierlich, stetig, differenzierbar
- Falls k von n fair, Rest 0, dann F(x) = k/n

CoNe Freiburg

Konvergenz

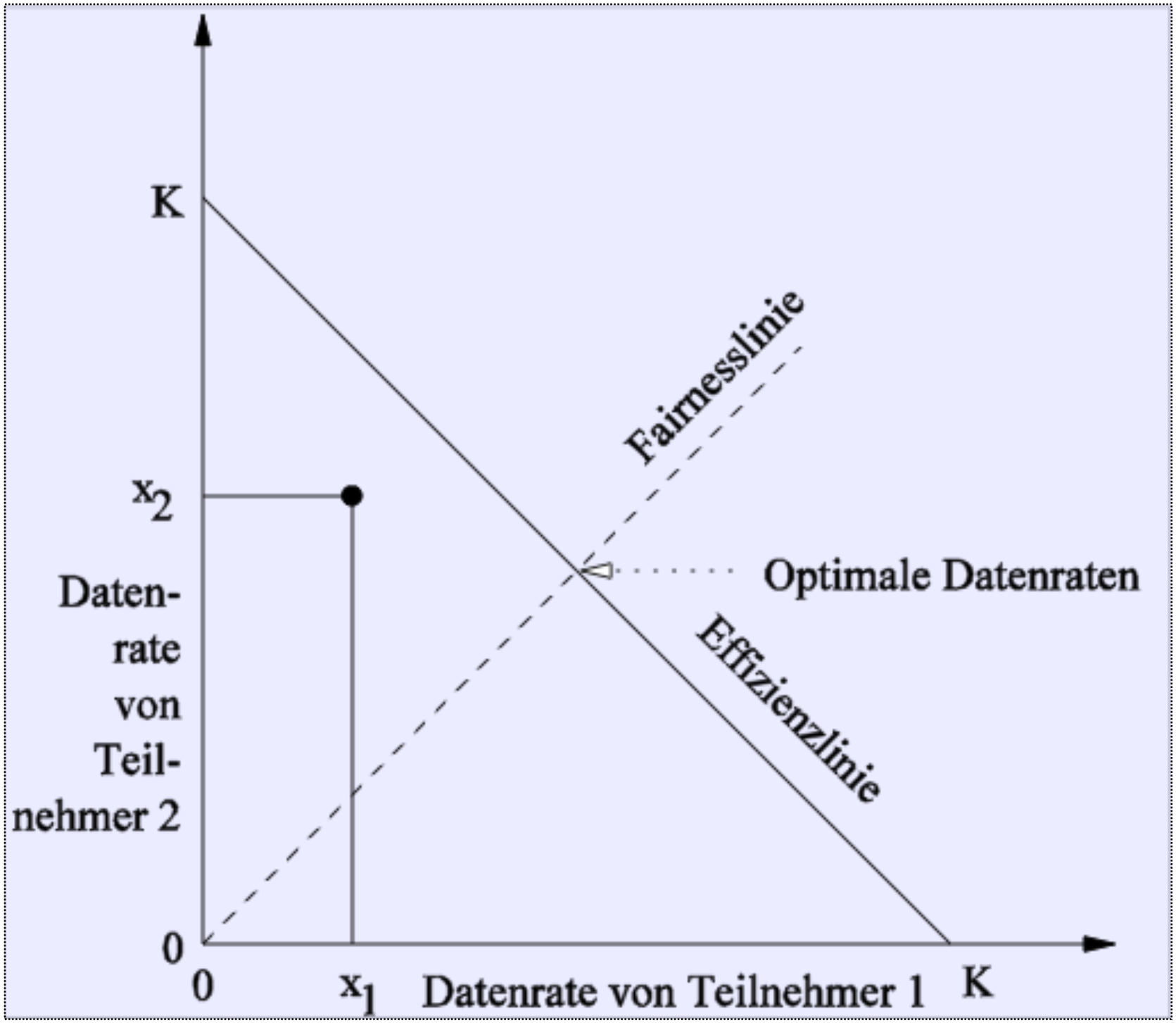
- Konvergenz unmöglich
- Bestenfalls Oszillation um Optimalwert
  - Oszillationsamplitude A
  - Einschwingzeit T



UNI FREIBURG

CoNe
Freiburg

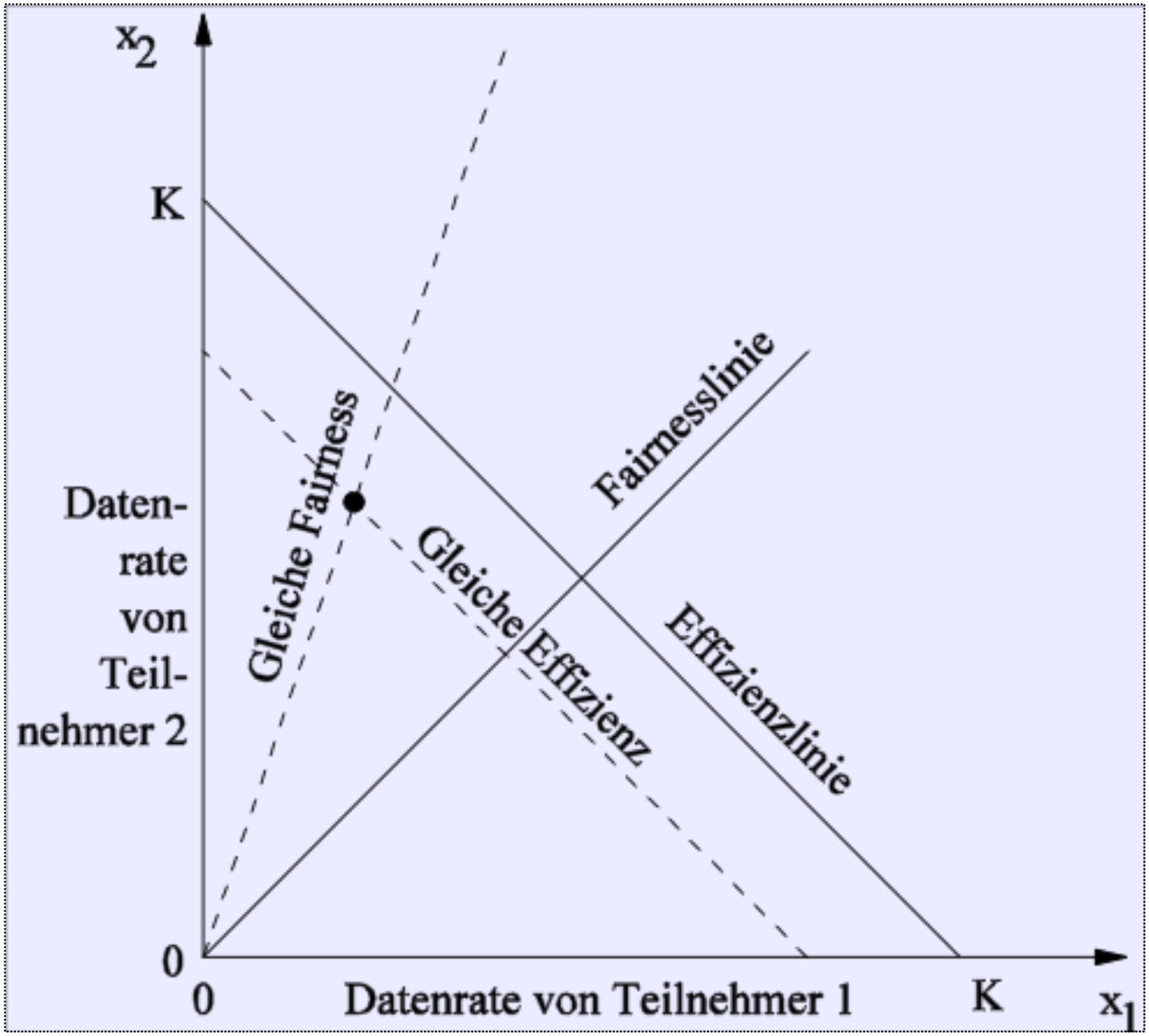
Vektordarstellung (I)



UNI FREIBURG

CoNe Freiburg

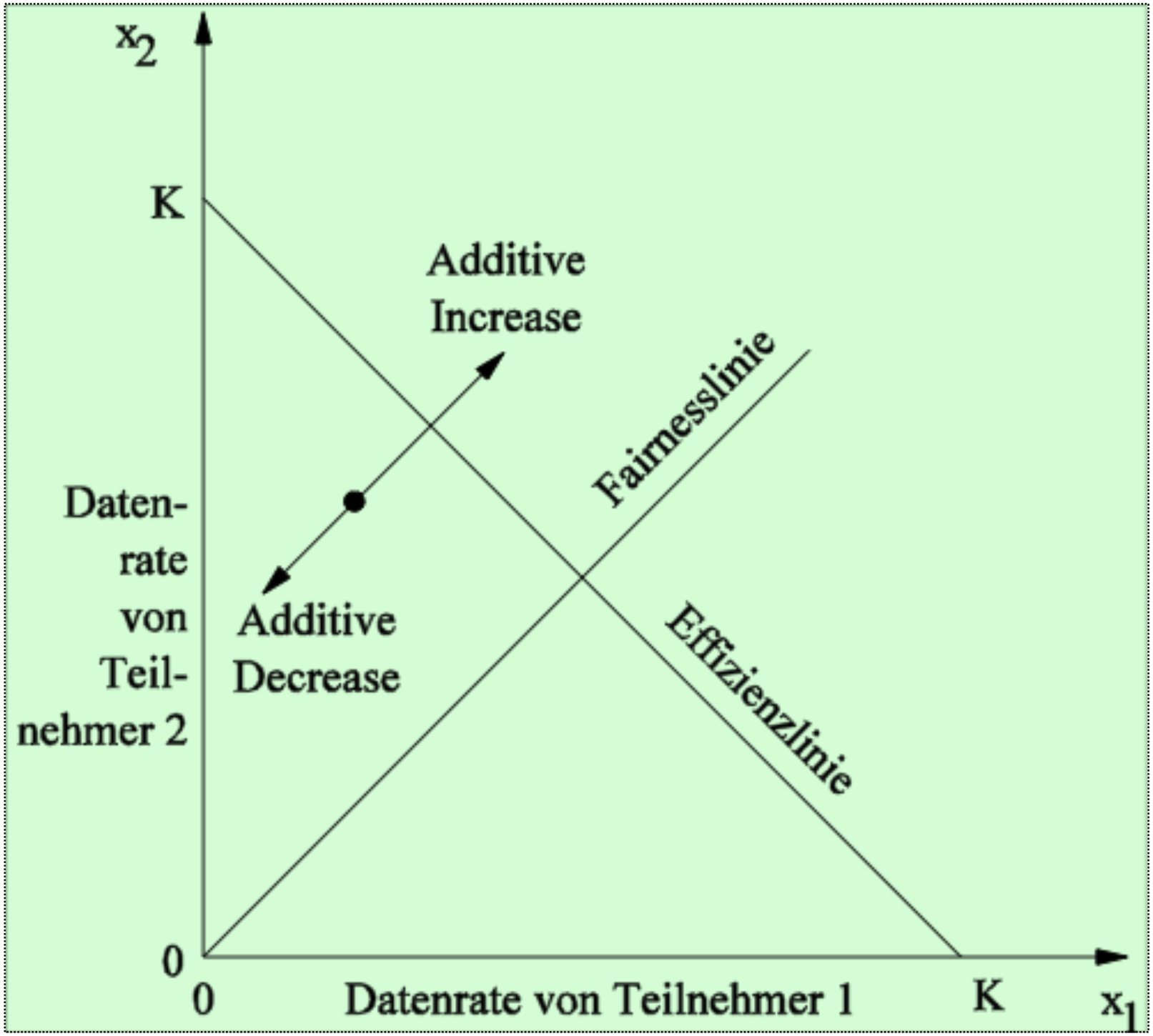
Vektordarstellung (II)



UNI FREIBURG

CoNe Freiburg

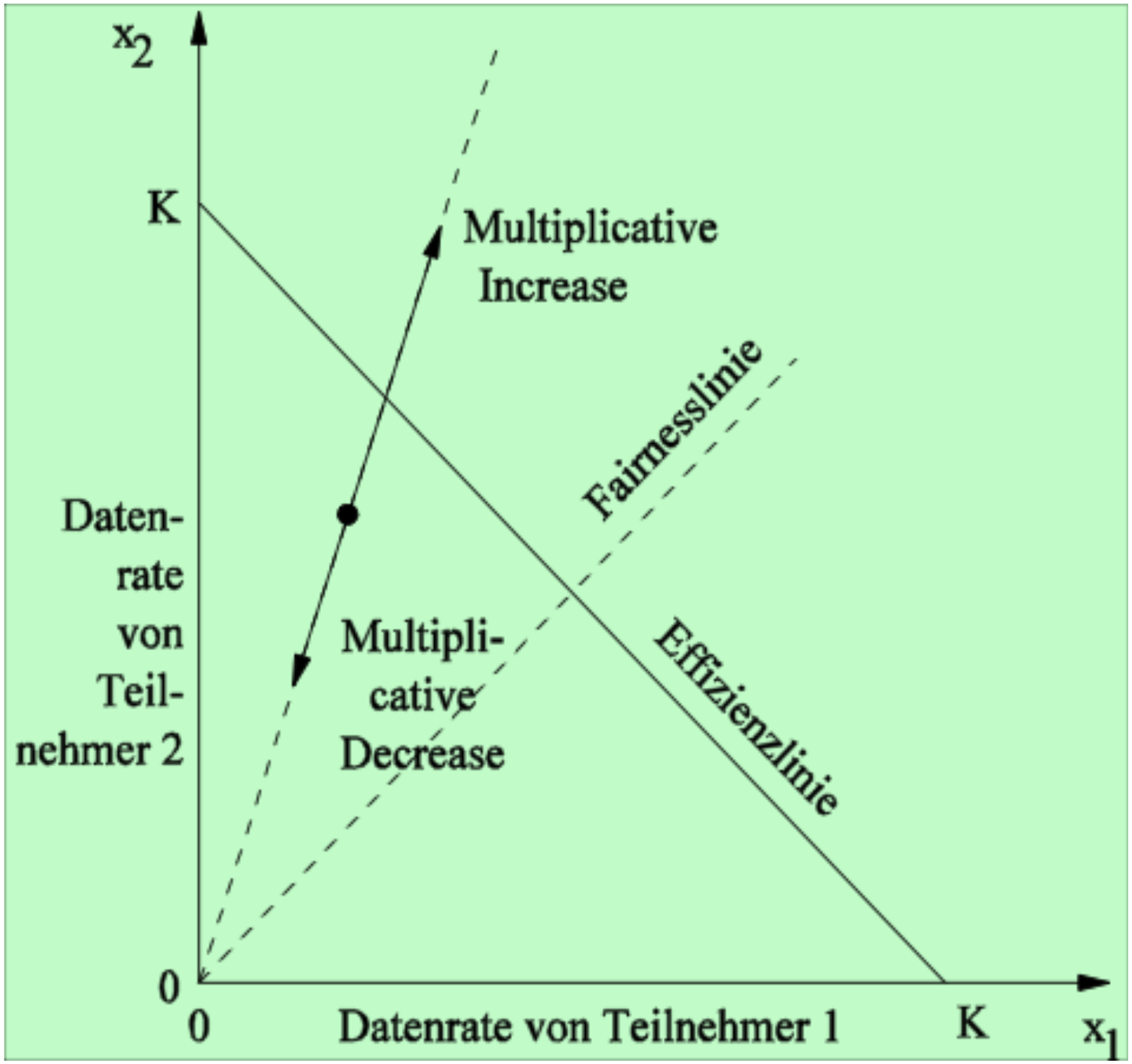
AIAD Additive Increase/ Additive Decrease



UNI FREIBURG

CoNe Freiburg

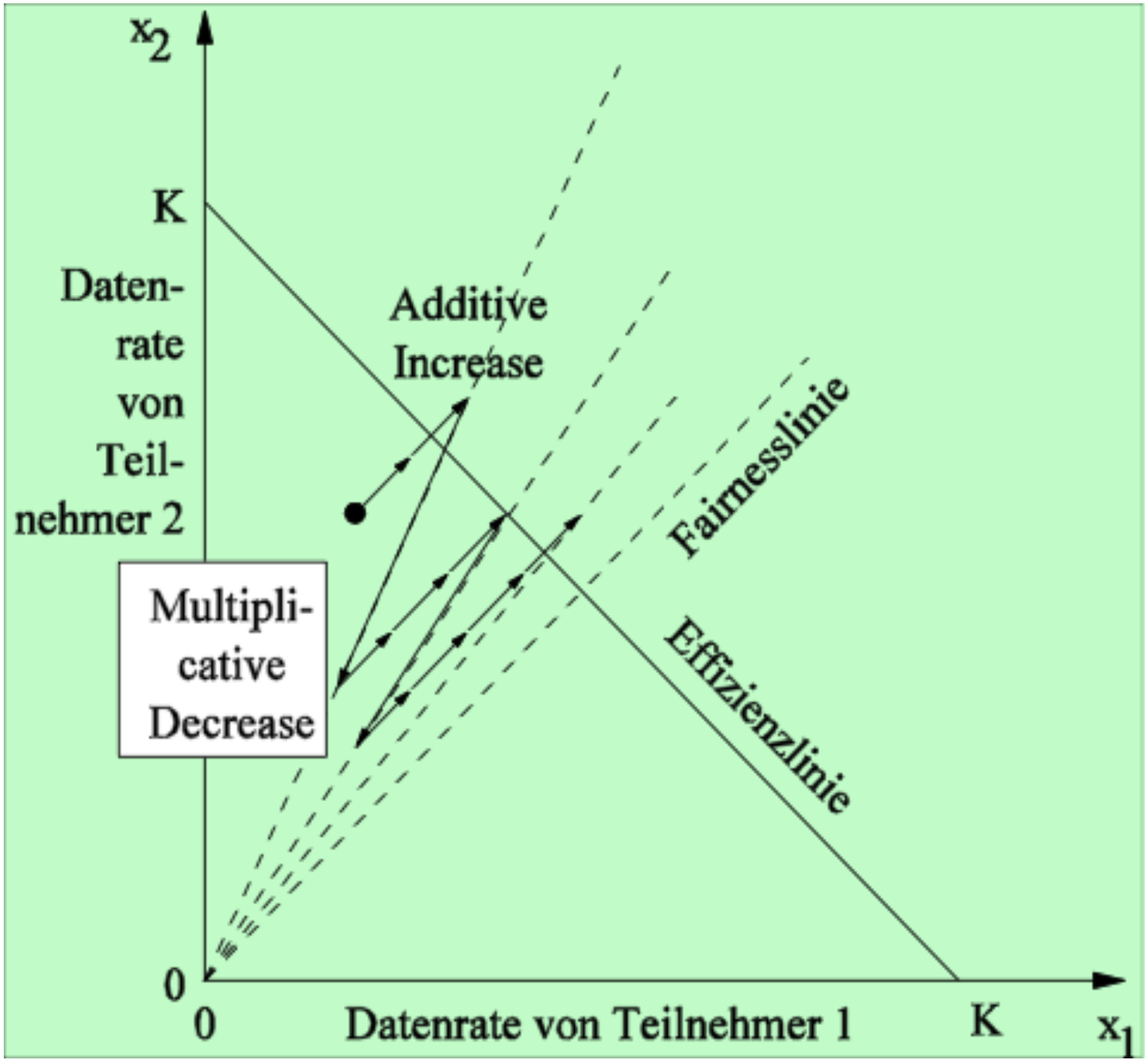
MIMD: Multiplicative Incr./ Multiplicative Decrease



UNI FREIBURG

CoNe Freiburg

AIMD: Additively Increase/ Multiplicatively Decrease



UNI FREIBURG

CoNe Freiburg

Probleme mit TCP Reno

- Verbindungen mit großer RTT werden diskriminiert
- Warum?
  - Auf jeden Router konkurrieren TCP-Verbindungen
  - Paketverluste halbieren Umsatz (MD)
  - Wer viele Router hat, endet mit sehr kleinen Congestion-Window
- Außerdem:
  - Kleinere RTT ist schnellere Update-Zeit
  - Daher steigt die Rate (AI) auf kurzen Verbindungen schneller
  - Mögliche Lösung:
    - konstante Datenratenanpassung statt Fenster-basierte Anpassung

CoNe Freiburg

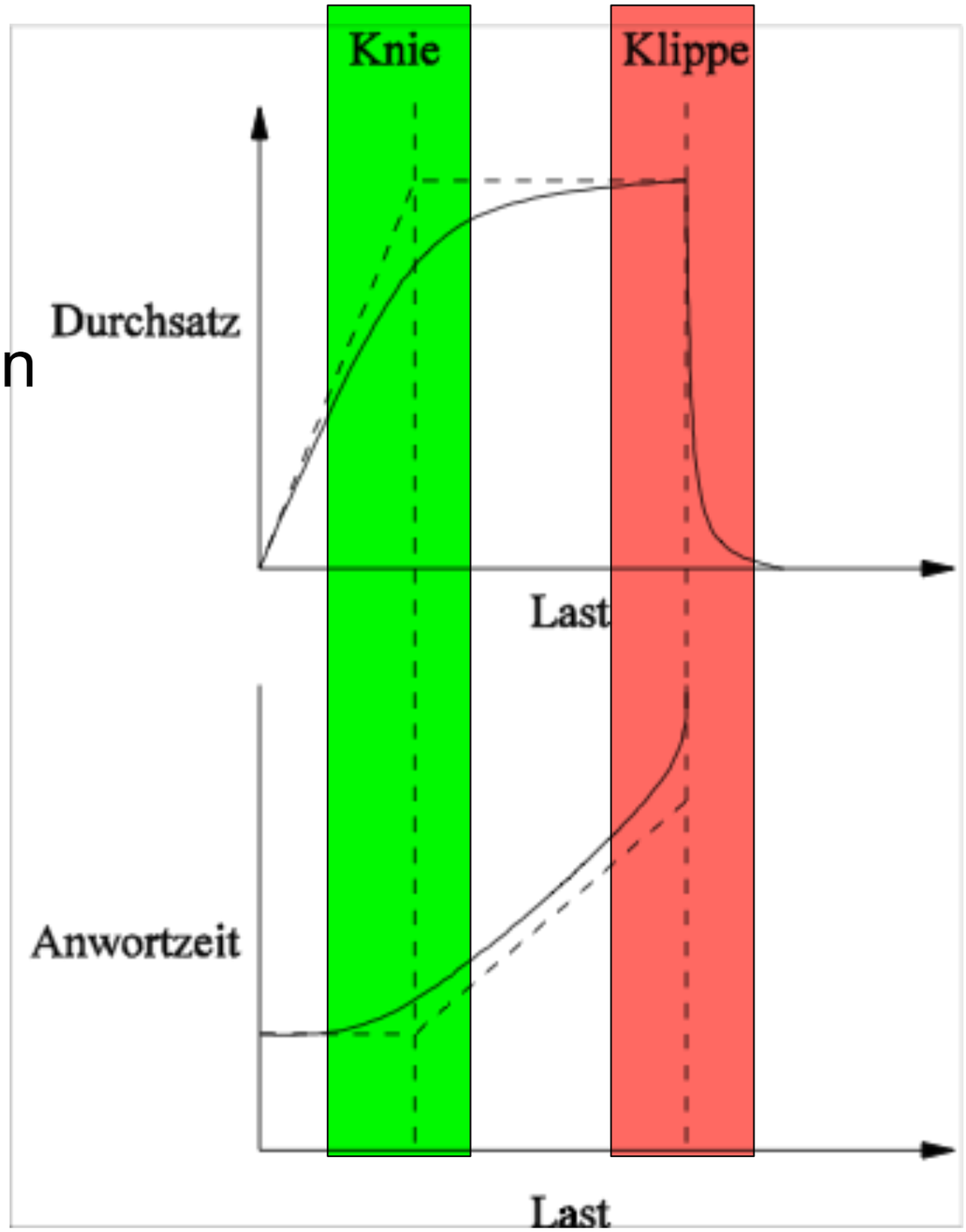
TCP Vegas

- RTT-basiertes Protokoll als Nachfolger von TCP Reno
  - “L. Brakmo and L. Peterson, “TCP Vegas: End-to-End Congestion Avoidance on a Global Internet”, IEEE Journal on Selected Areas of Communications, vol. 13, no. 8, October 1995, pp. 1465–1480.
- Bessere Effizienz
- Geringere Paketverluste
- Aber:
  - TCP Vegas und TCP Reno gegeneinander unfair

CoNe Freiburg

Durchsatz und Antwortzeit

- Klippe:
  - Hohe Last
  - Geringer Durchsatz
  - Praktisch alle Daten gehen verloren
- Knie:
  - Hohe Last
  - Hoher Durchsatz
  - Einzelne Daten gehen verloren



UNI FREIBURG

CoNe
Freiburg

TCP Vegas-Algorithmus

UNI FREIBURG

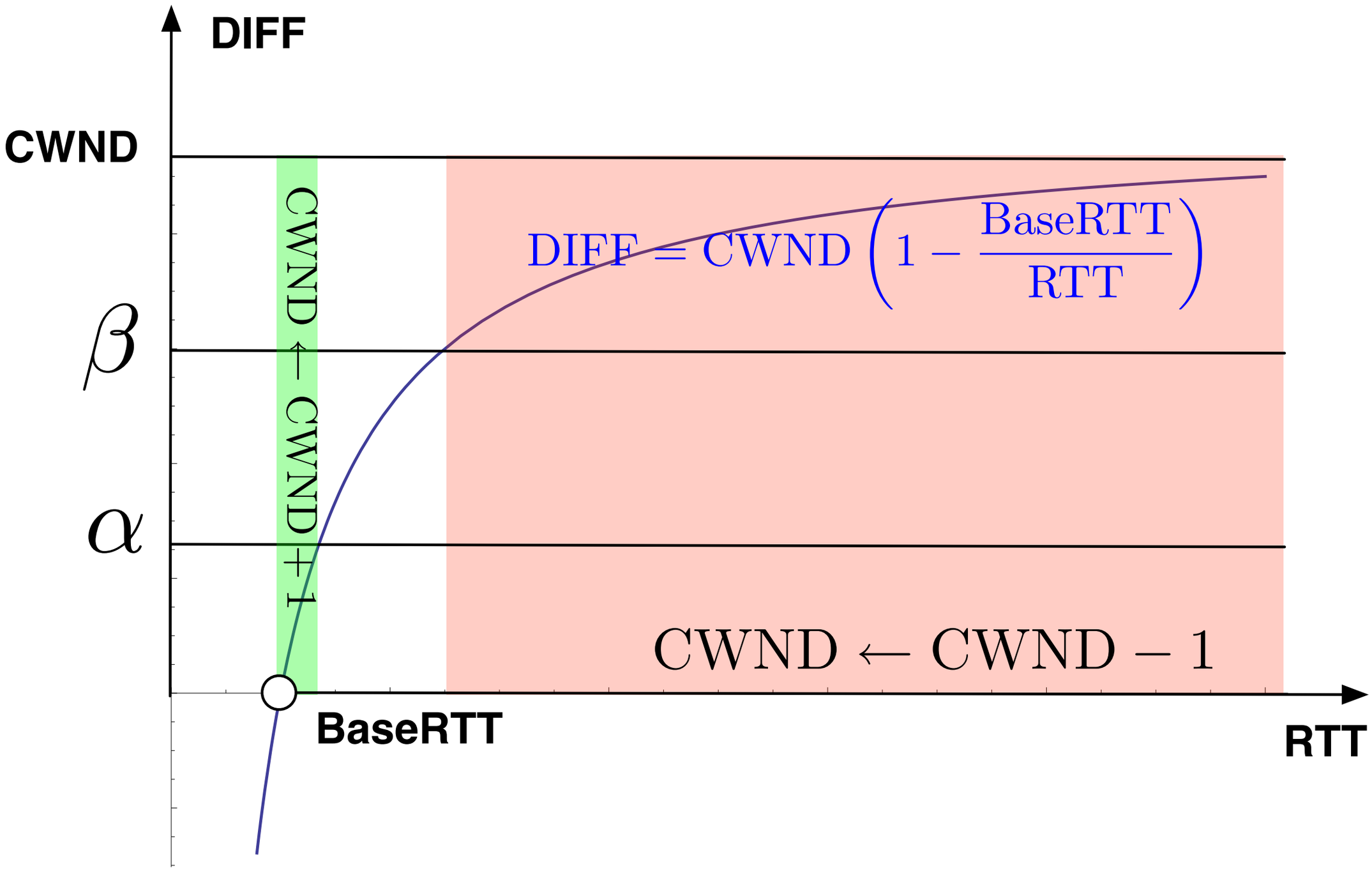
CoNe Freiburg

TCP Vegas-Algorithmus

- Parameter
  - geschätzte Umlaufzeit: RTT
  - minimale Umlaufzeit: $BaseRTT$
  - wirkliche Datenrate: $Actual = CWND/RTT$
  - erwartete Datenrate: $Expected = CWND/BaseRTT$
  - $Diff = (Expected - Actual)\ BaseRTT$
  - $Programmparameter: 0 \leq \alpha < \beta$
- Wenn $Diff \leq \alpha$ (d.h. $Actual \approx Expected$)
  - Last ist gering
  - $CWND \leftarrow CWND + 1$
- Wenn $Diff > \beta$, (d.h. $Actual << Expected$)
  - Last ist zu hoch
  - $CWND \leftarrow CWND - 1$
- Sonst keine Aktion: $CWND \leftarrow CWND$

CoNe Freiburg

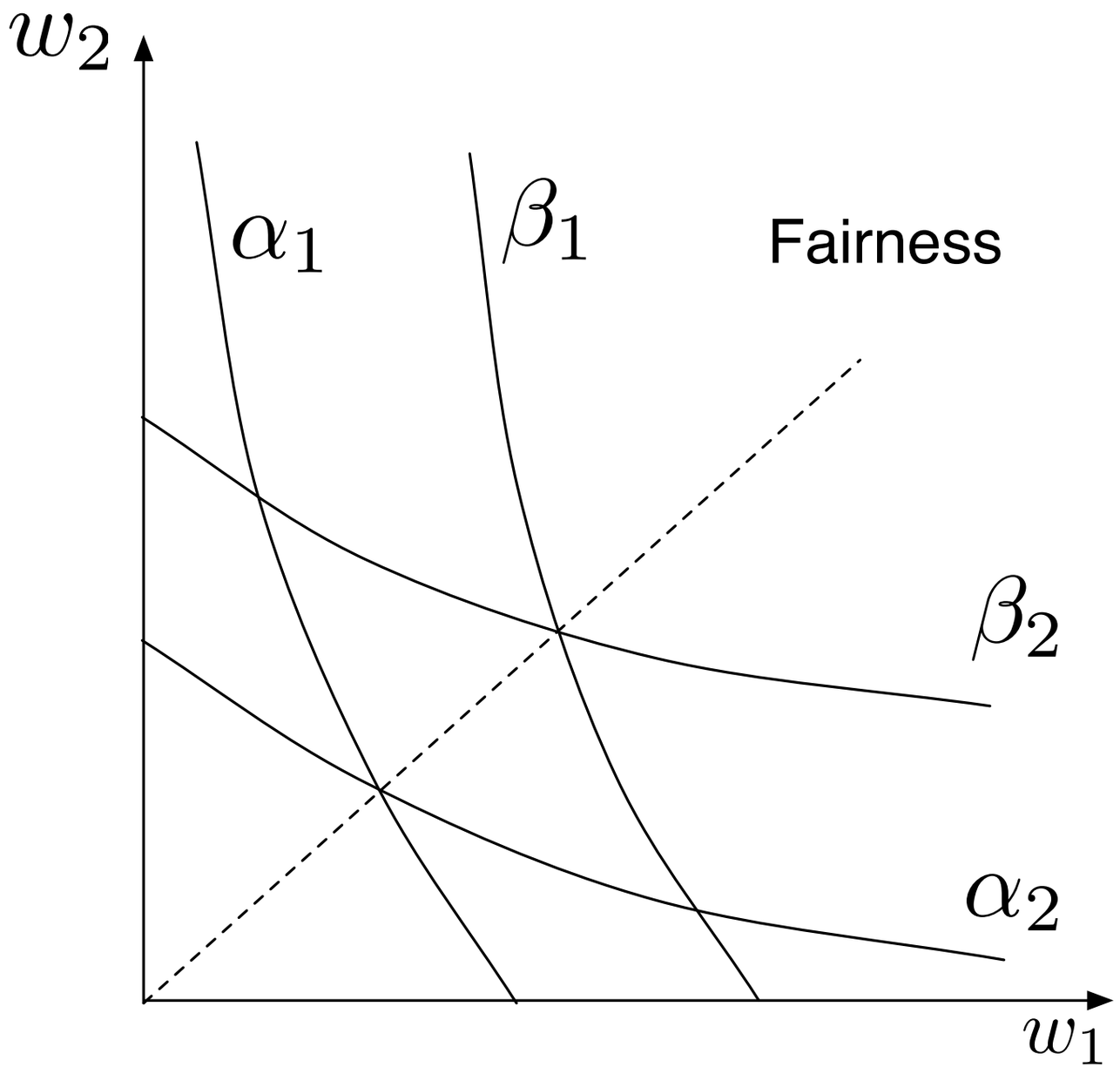
TCP Vegas - Abhängigkeit von RTT



UNI FREIBURG

CoNe Freiburg

Fenster-Anpassung in Vegas



UNI FREIBURG

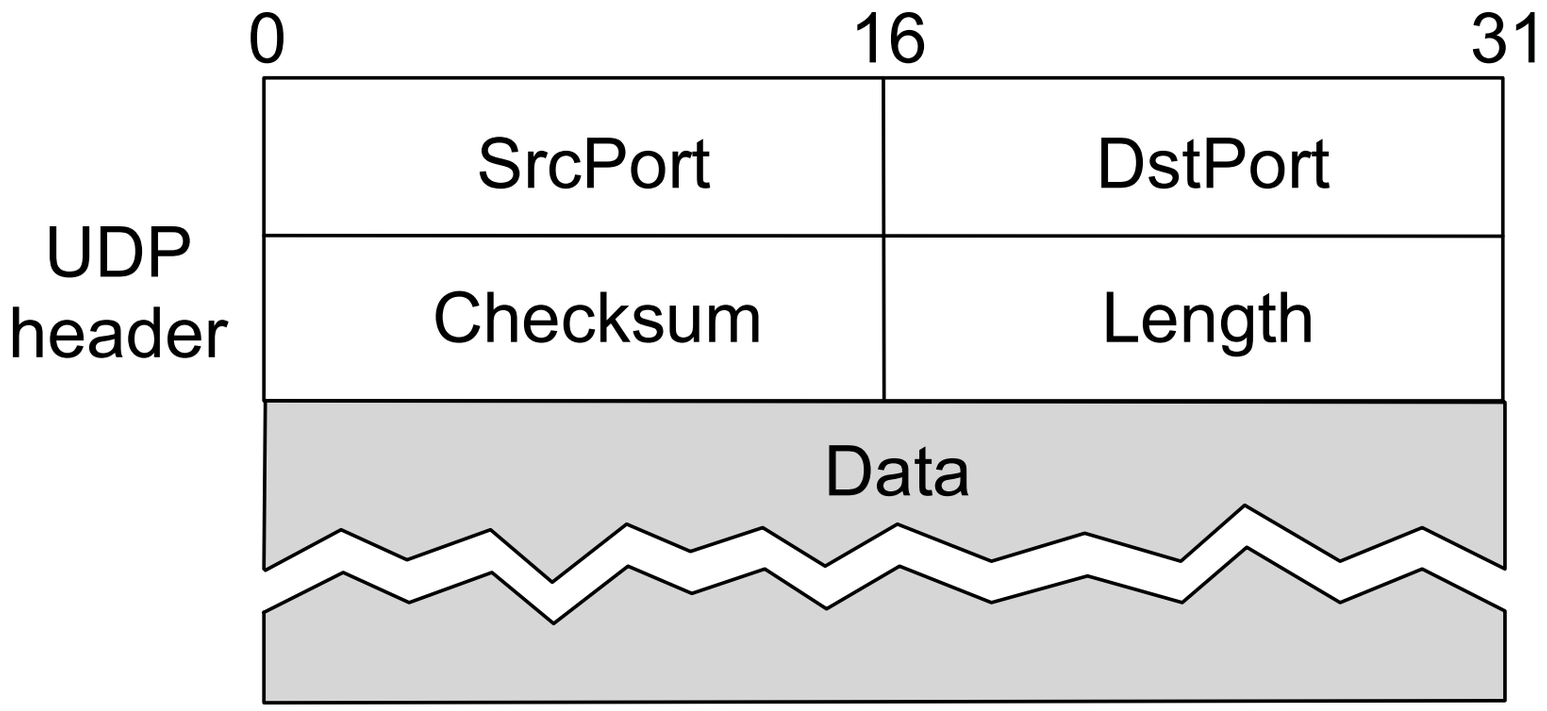
CoNe Freiburg

TCP Fairness & TCP Friendliness

- TCP
  - reagiert dynamisch auf die zur Verfügung stehende Bandweite
  - Faire Aufteilung der Bandweite
    - Im Idealfall: n TCP-Verbindungen erhalten einen Anteil von 1/n
- Zusammenspiel mit anderen Protokollen
  - Reaktion hängt von der Last anderer Transportprotokolle ab
    - z.B. UDP hat keine Congestion Control
  - Andere Protokolle können jeder Zeit eingesetzt werden
  - UDP und andere Protokoll können TCP Verbindungen unterdrücken
- Schlussfolgerung
  - Transport-Protokolle müssen TCP-kompatibel sein (TCP friendly)

UDP

- User Datagram Protocol (UDP)
  - ist ein unzuverlässiges, verbindungsloses Transportprotokoll für Pakete
- Hauptfunktion:
  - Demultiplexing von Paketen aus der Vermittlungsschicht
- Zusätzlich (optional):
  - Checksum aus UDP Header + Daten



CoNe Freiburg

TCP Zusammenfassung

- TCP erzeugt zuverlässigen Byte-Strom
  - Fehlerkontrolle durch “GoBack-N”
- Congestion control
  - Fensterbasiert
  - AIMD, Slow start, *Congestion Threshold*
  - Flusskontrolle durch *Window*
  - Verbindungsaufbau
  - Algorithmus von Nagle

CoNe Freiburg

IIF INSTITUT FÜR INFORMATIK FREIBURG

Systeme II

5. Die Transportschicht

Thomas Janson°, Kristof Van Laerhoven*, Christian Ortolf°

Folien: Christian Schindelhauer°

Technische Fakultät

°: Rechnernetze und Telematik, *: Eingebettete Systeme

Albert-Ludwigs-Universität Freiburg

Version 24.04.2015

UNI FREIBURG