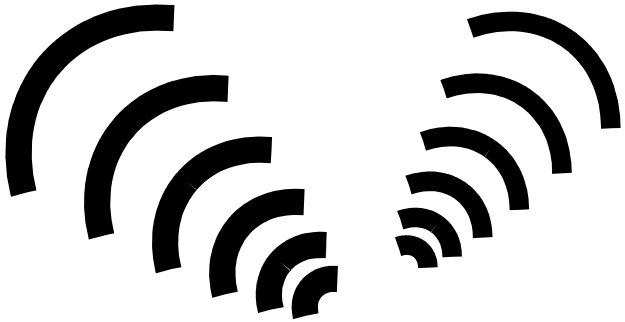


TCP with Adaptive Pacing im Linux-Kernel

Erhöhung von Durchsatz und Fairness in drahtlos vermaschten Netzen

Dipl.Inf. Sebastian Meiling*

6. Oktober 2009

** Auf Basis von Forschungsarbeiten von Sherif M. ElRakabawy et al.*

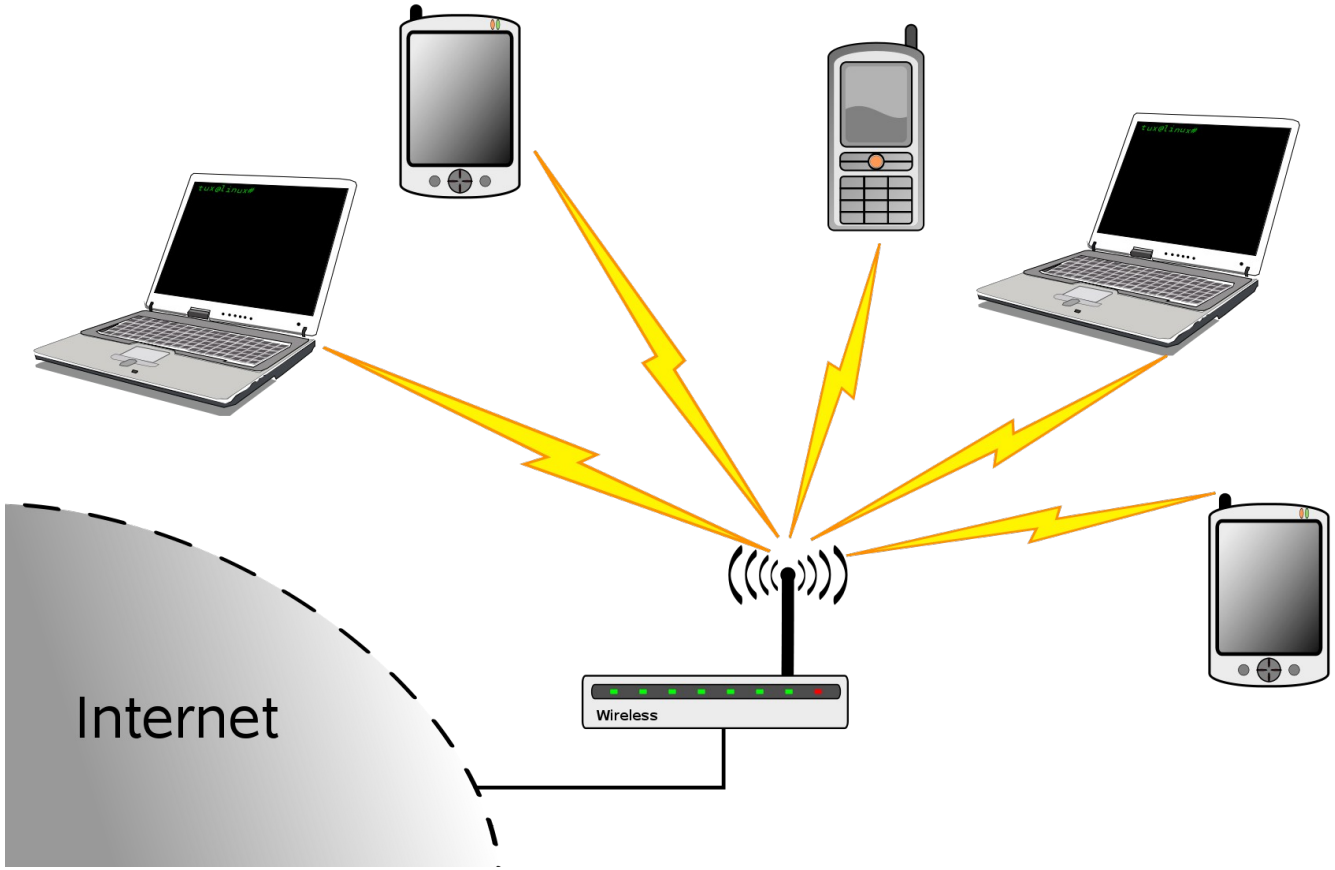
Gliederung des Vortrags

- Einleitung
- TCP-AP Algorithmus
- Implementierung
- Experimente
- Zusammenfassung
- Literatur

Einleitung

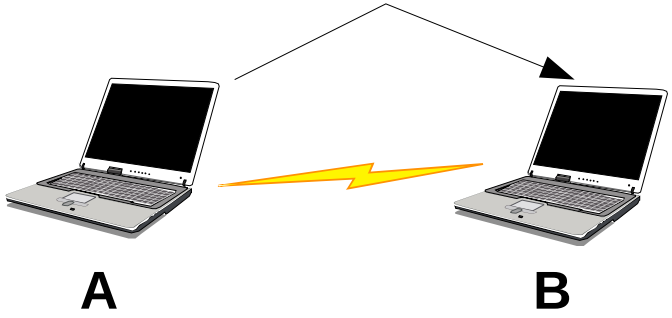
Eigenschaften drahtloser Netze

Heutige drahtlose Netzwerke

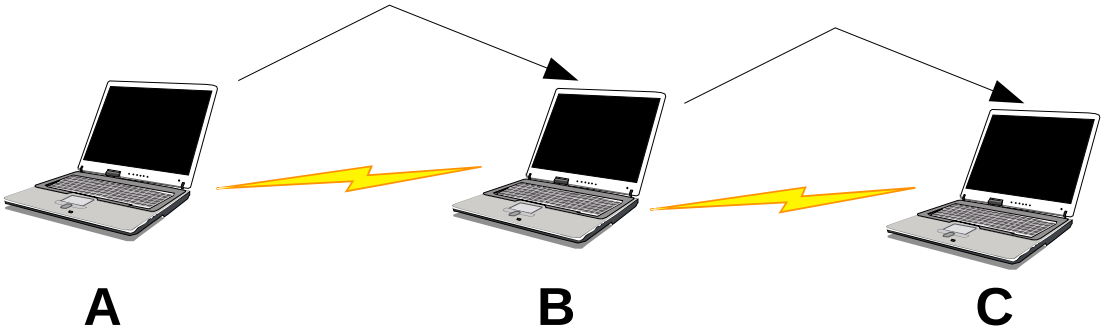


Drahtlose Ad-Hoc-Verbindungen

- Einfache Ad-Hoc Verbindung (A nach B)



- Multi-hop Verbindung (A nach C)

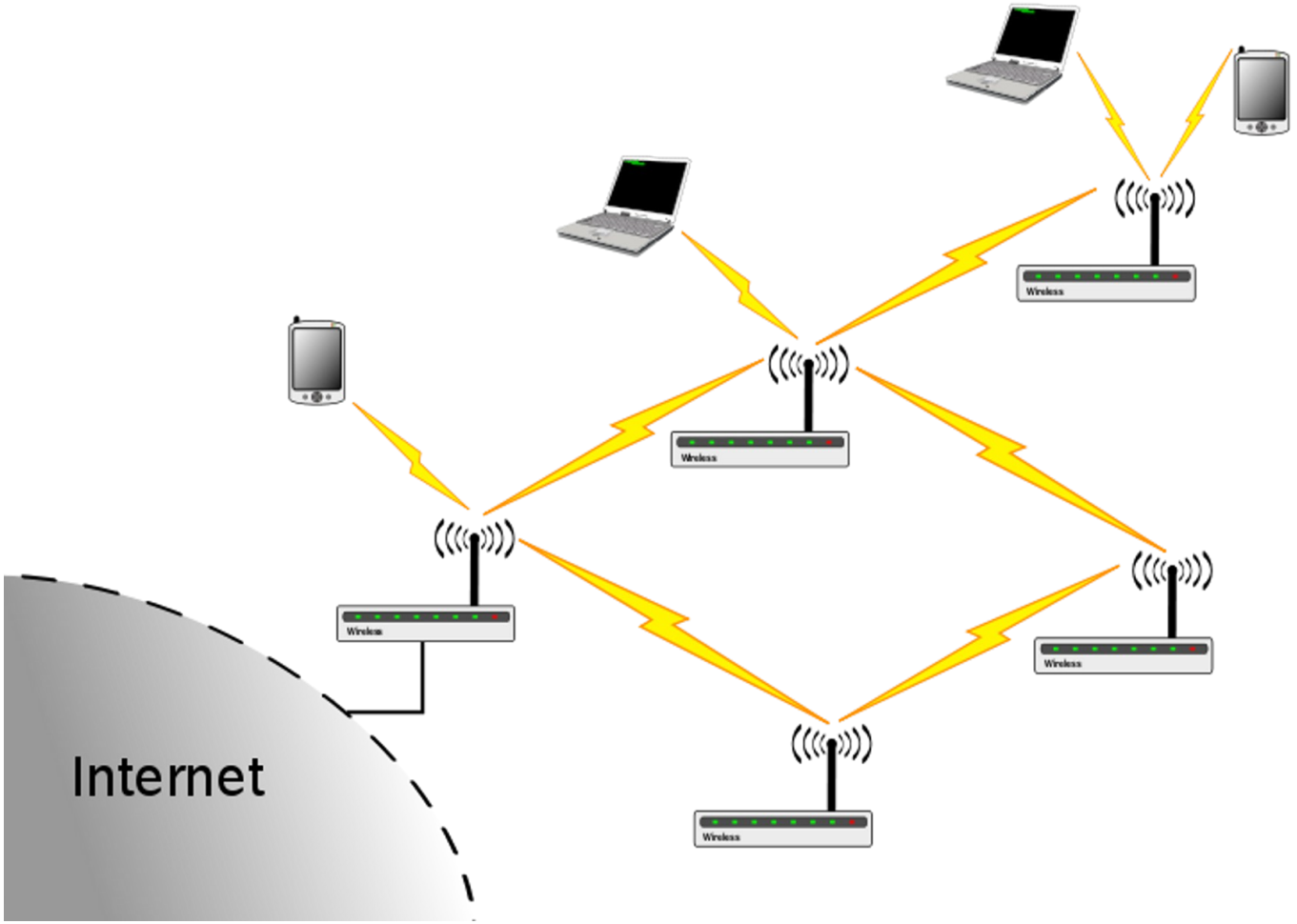


Eigenschaften und Verwendung

- Direkte Verbindung zwischen Endgeräten ohne Access-Point
- Zwischenknoten als Relay bei Multi-Hop

- Car-to-Car Kommunikation (VANET)
- Pocket-Switched Networks
- Opportunistische Erweiterung des Internets

Drahtlos vermaschte Netze



Eigenschaften und Verwendung

- Drahtlose ad-hoc Verbindungen zwischen den Netzteilnehmern
- Dezentrale Organisation, ohne klassische Infrastruktur-Punkte
- Gute Lastverteilung und Fehlertoleranz möglich

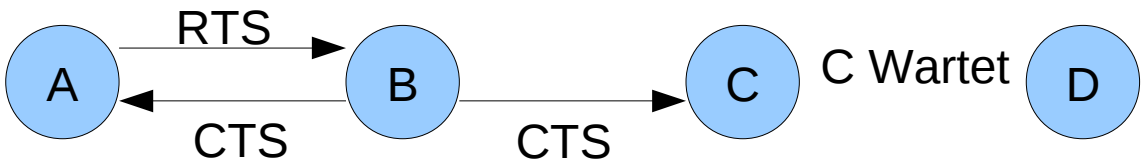
- Versorgung von Regionen ohne DSL-Zugang mit Breitband-Internet (z.B. *Freifunk-Netz*)

Eigenschaften drahtloser Übertragungen

- Luft als Übertragungsmedium
- Übertragungsreichweite < Interferenzreichweite

- Hidden-Terminal-Effekt: A-B stört C-D

- Lösung: RTS-CTS ?



TCP in drahtlosen Netzen

- Fenster basierte Datenübertragung
- „provoziert“ Paketverluste um verfügbare Bandbreite zu erkennen
- Backoff auf MAC-Schicht kann zu Unfairness konkurrierender TCP-Übertragungen führen
- Aufeinander folgende Bestätigungspakete (ACKs) bewirken Packet-Burst

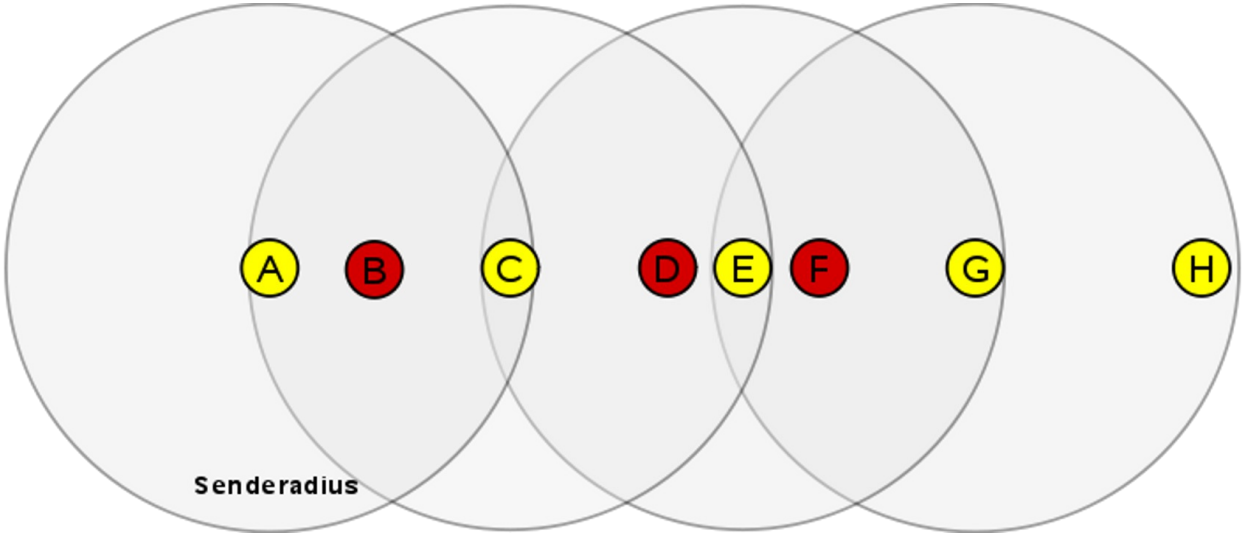
TCP-AP Algorithmus

Funktionsweise des Adaptive Pacing Verfahrens

Eigenschaften von TCP-AP

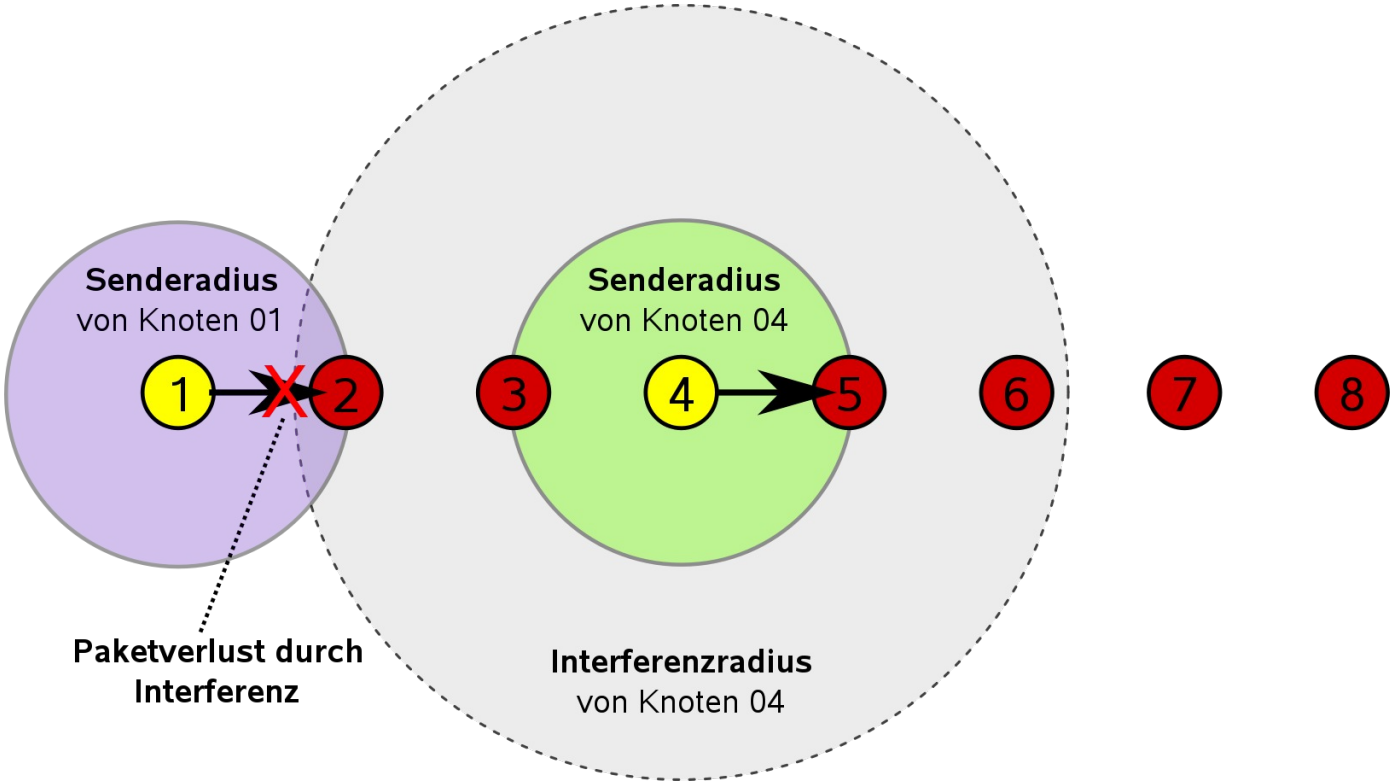
- Verfahren zur Fluss- und Überlastkontrolle, aber ohne direkten Einfluss auf das Überlastfenster
- Timer-gesteuertes, raten basiertes Versenden von Paketen
- Bestimmung des (optimalen) Sende-Intervalls durch lokale Information am Sender
- Nur Änderungen am Sender erforderlich
- TCP-kompatibles/freundliches Verhalten
- Vorteil: inkrementelle Verbreitung und Einsatz parallel zu Standard-TCP möglich

Annahmen

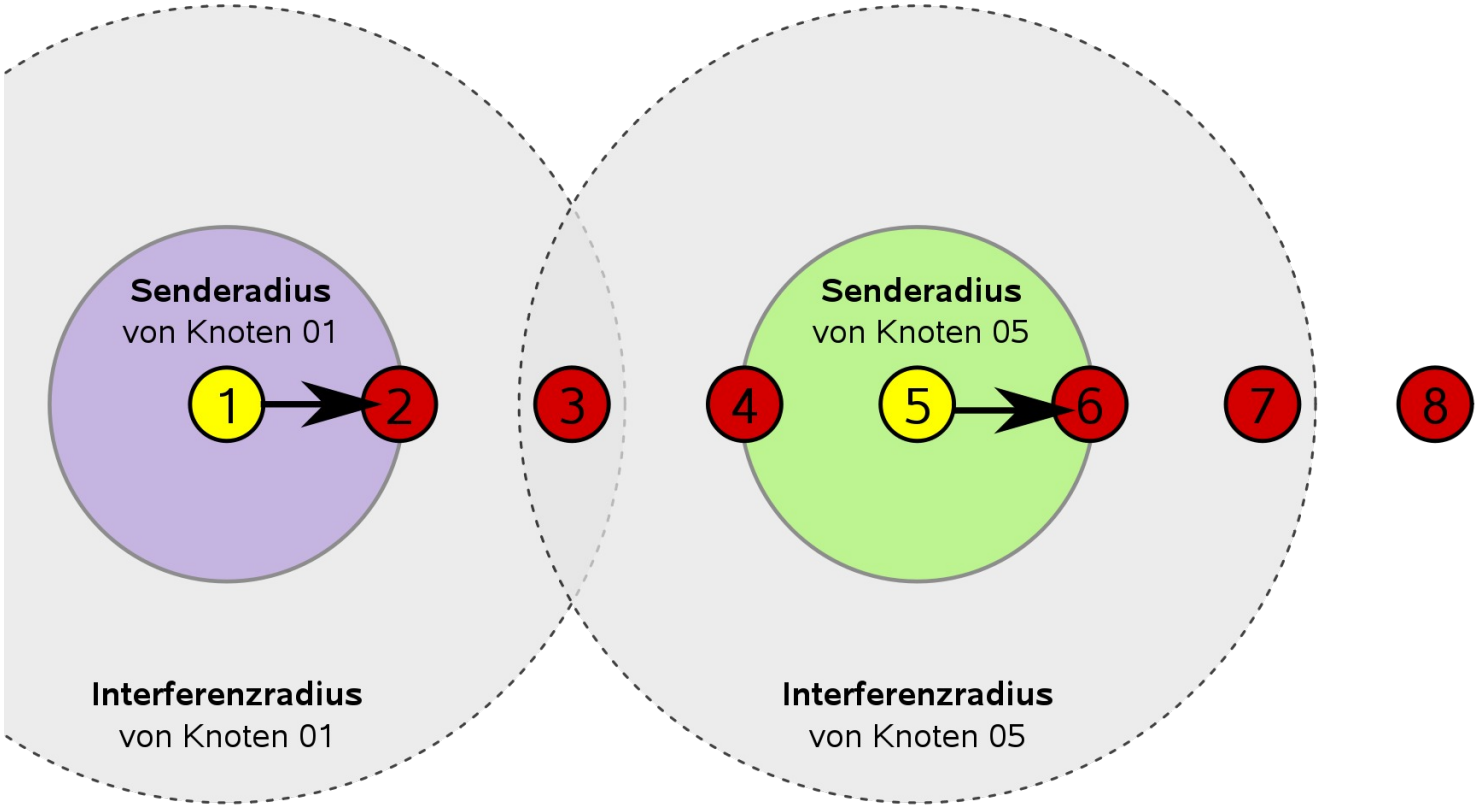


- Routingprotokoll wählt kürzesten Pfad
- Knoten sind etwa gleich weit voneinander entfernt
- Ähnliche Bedingungen entlang des Pfades

Problem: Interferenz bei Übertragung



Lösung: Verzögerung der Übertragung



Informationen am Sender

- Transport-Schicht:
  - Round Trip Time (RTT)
  - Paketgrößen
- Routing-Schicht:
  - Anzahl der Hops vom Sender zum Empfänger
- MAC-Schicht (WLAN-Treiber):
  - Sendeleistung
  - Empfangsleistung

Berechnung des Sende-Intervalls

- Optimales Intervall abhängig von Last im Netz und Interferenz-Reichweite
- Einflussgrößen
  - Variationskoeffizient (COV_{RTT})
  - Interfenz-Verzögerung (OID)

- COV_{RTT} ist Maß für die aktuelle Netzlast
- OID approximiert die Interferenz-Verzögerung, auf Basis von Treiberinformationen

Der Variationskoeffizient

- hohe Netzlast führt zu Schwankungen der RTT
- *Problem:* starke Änderung des Sende-Intervalls unerwünscht
- *Lösung:* Einführung eines Glättungsfaktors
- Berechnung des Variationskoeffizienten über die N letzten RTT-Werte

$$COV_{RTT} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(RTT_i - \overline{RTT})^2}}{\overline{RTT}}$$

Berechnung der Interferenz-Verzögerung

- Interferenz-Reichweite H in Hops:

$$H = min(n; n \in \{1, 2, ... k\} \wedge P_{out} - L_n \leqslant T_{cs})$$

- k = Pfadlänge
- P_{out} = Reale Ausgangsleistung
- L_n = Signaldämpfung nach n Hops
- T_{cs} = Carrier Sensing Threshold

Interferenz-Verzögerung

- Zusammensetzung der RTT:

$$RTT = k \cdot (t_q + t_{data}) + k \cdot (t_q + t_{ack})$$

- k = Pfadlänge
- t_q = mittleres Queuing-Delay
- t_{data} = Übertragungszeit für Datenpaket
- t_{ack} = Übertragungszeit für ACK-Paket

Interferenz-Verzögerung

- Interferenz-Verzögerung OID ist:

$$OID = (H+1) \cdot \left(t_q + \frac{s_{data}}{b}\right)$$

- OID = Out of Interference Delay
- t_q = Queuing-Delay
- s_{data} = Größe eines Datenpaket
- b = Übertragungsrate (in Byte/s)

Berechnung des Sende-Intervalls

- Verwendung des gewichteten Mittelwertes der Interferenz-Verzögerung (mit $\alpha = 0{,}7$):

$$\widehat{OID} = \alpha \cdot OID_{alt} + (1-\alpha)\, OID_{neu}$$

- Sende-Intervall S berechnet sich als:

$$S = (1 + 2 \cdot COV_{RTT}) \cdot \widehat{OID}$$

TCP-AP Algorithmus (Pseudo-Code)


```
FUNC tcp_receive ()
{
        For each Empfangenes ACK do {
                Approximiere Interferenzreichweite in Hops;
                Berechne Variationskoeffizienten über N RTTs;
                Berechne Interferenzverzögerung;
                Setze neues Sende-Intervall;
        }
}

FUNC pacing_timeout()
{
        If weitere Pakete then
                Sende nächstes Paket;
        Else
                Warte;

        Starte Timer neu;
}
```

Implementierung

Besonderheiten der Kernel-Realisierung

Besonderheiten des Linux-Kernels

- Eingeschränkte mathematische Funktionen
  - Nur ganzzahlige Operationen
  - d.h. keine Wurzelfunktion

- Fester innerer Takt des Kernels, daher eingeschränkte Genauigkeit bei Timern
  - Standard: 250Hz, entspricht 4ms
  - Für TCP-AP: 1000Hz, entspricht 1ms

Anpassung des Variationskoeffizienten

- *Problem:* Berechnung der Standardabweichung[1] nicht möglich
- *Lösung:* Verwendung der mittleren absoluten Abweichung[2]

$$MAD_{RTT} = \frac{\frac{1}{N}\sum_{i=1}^{N} |RTT_i - \overline{RTT}|}{\overline{RTT}}$$

- Man kann zeigen: $MAD_{RTT} \approx 0{,}8 \cdot COV_{RTT}$

1 standard deviation

2 mean absolute deviation

Berechnung des Sende-Intervalls

- *Problem:* Nur Millisekunden-genauer Timer für Sende-Intervall möglich
- *Lösung:* Aufaddieren der Nachkommastellen
- *Beispiel:*
  - Berechnetes Sendeintervall: S' = 2,5ms
  - Tatsächliches Sendeintervall: S = 2ms
  - Rest: R = 0,5
  - Betrachte: 1ms / 0,5ms = 2, d.h. Erhöhe S bei jedem zweiten Paket
  - Ergibt: S_1 = 2ms, S_2 = 3ms, S_3 = 2ms usw.

Experimente

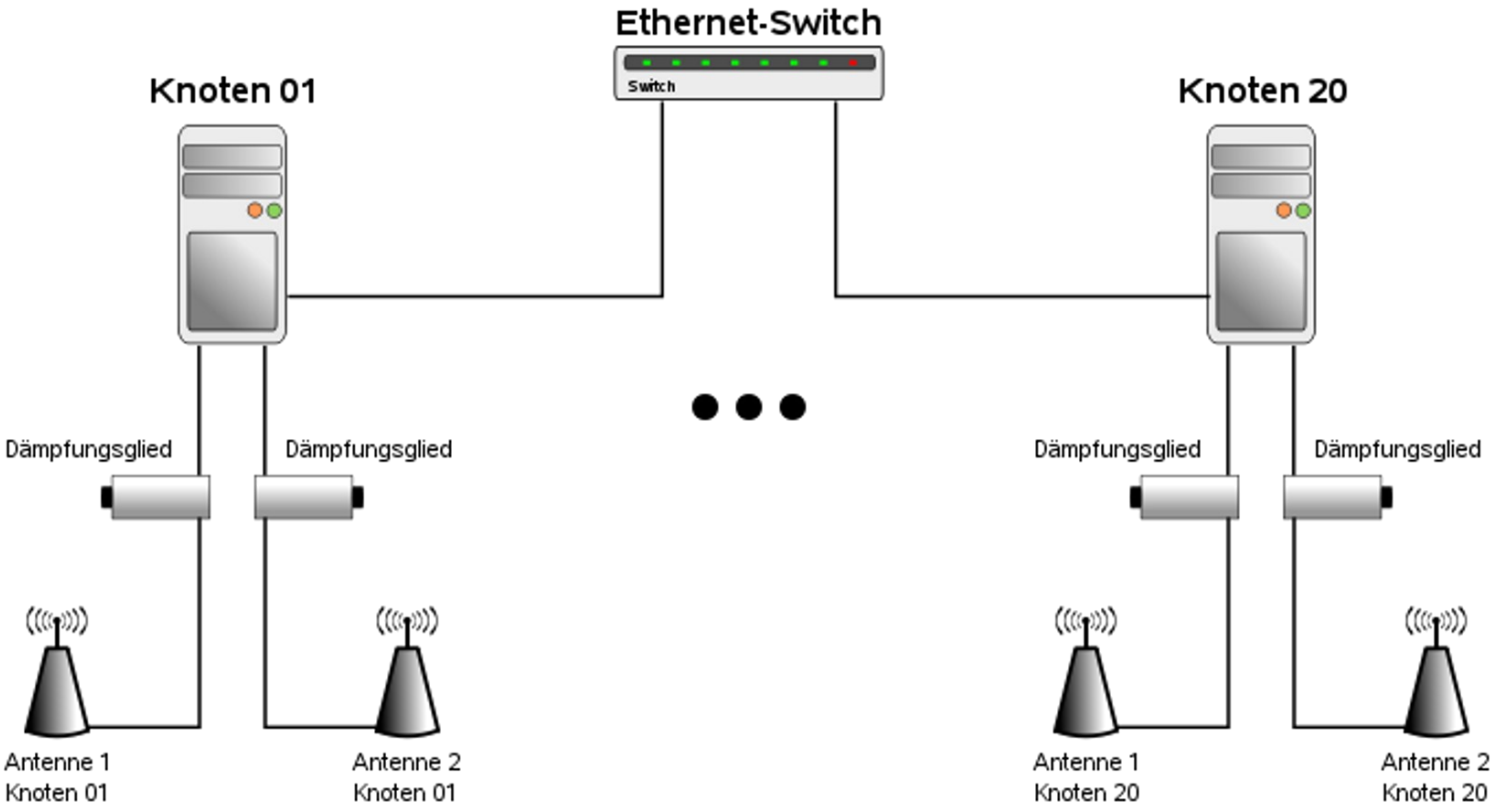
Aufbau, Durchführung und Ergebnisse

Das Miniatur Testbed

- 20 Desktop-Rechner mit Standard-Hardware
- je 2 WLAN-Karten (802.11 b/g) pro Knoten
- Variable Netzwerk-Topologie durch frei aufstellbare Rundstrahl-Antennen
- Reduzierung der Signalstärke durch manuelle Dämpfungsglieder
- Konfiguration und Überwachung über Ethernet-Schnittstelle

Das Miniatur Testbed

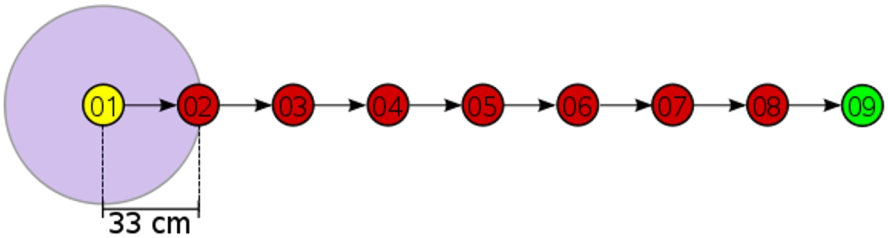
Schema des Testbeds



Durchführung der Experimente

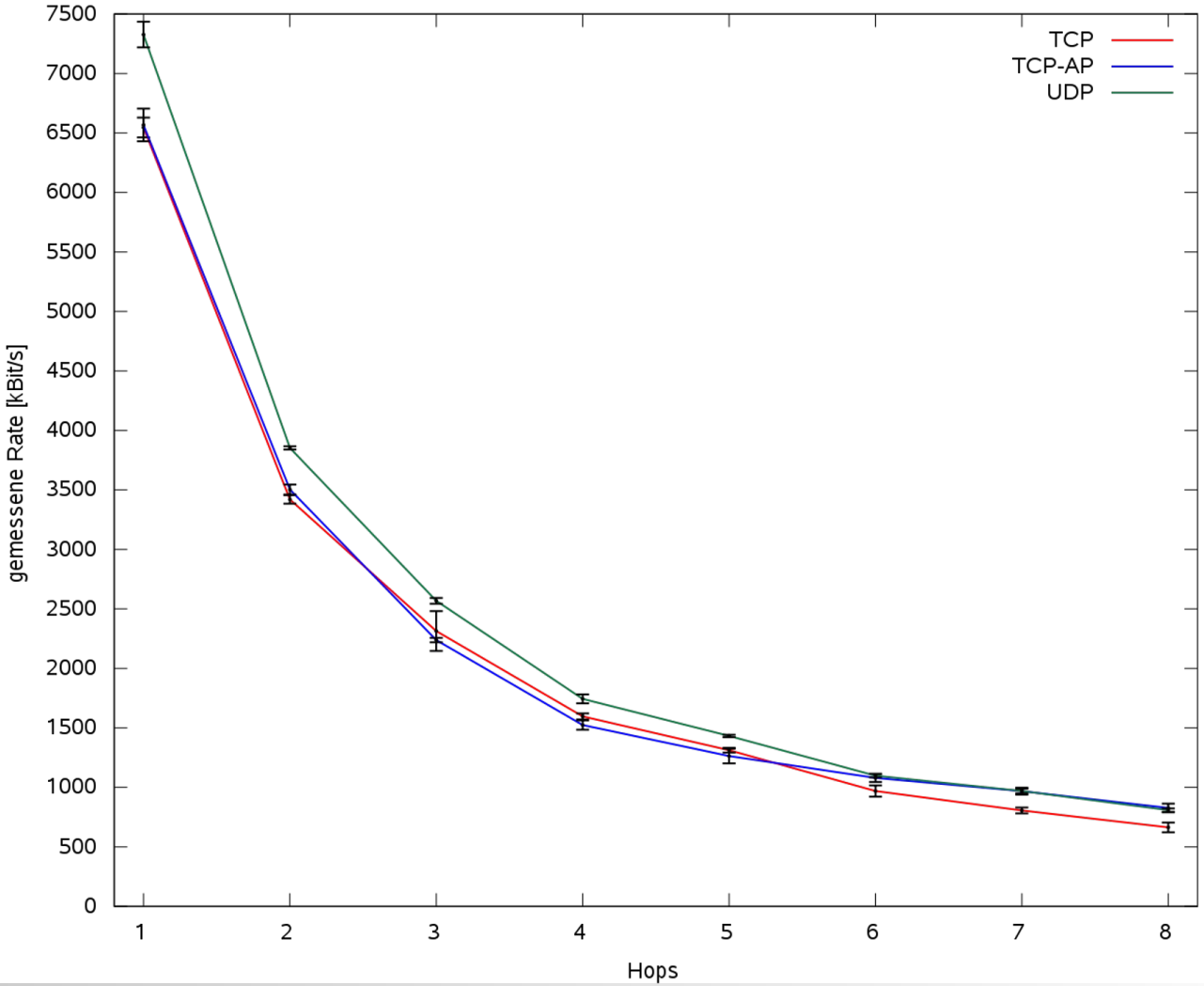
- Verwendung einfacher Kettentopologien
- Einrichtung statischer Routen zwischen den Knoten
- Knoten können nur direkte Nachbarn erreichen
- Datenübertragung wird mit *Iperf* generiert und aufgezeichnet
- Auswertung des Goodputs am Empfänger
- Laufzeit jeweils 40s und gemittelt über 20 Durchläufe

1. Experiment: Aufbau

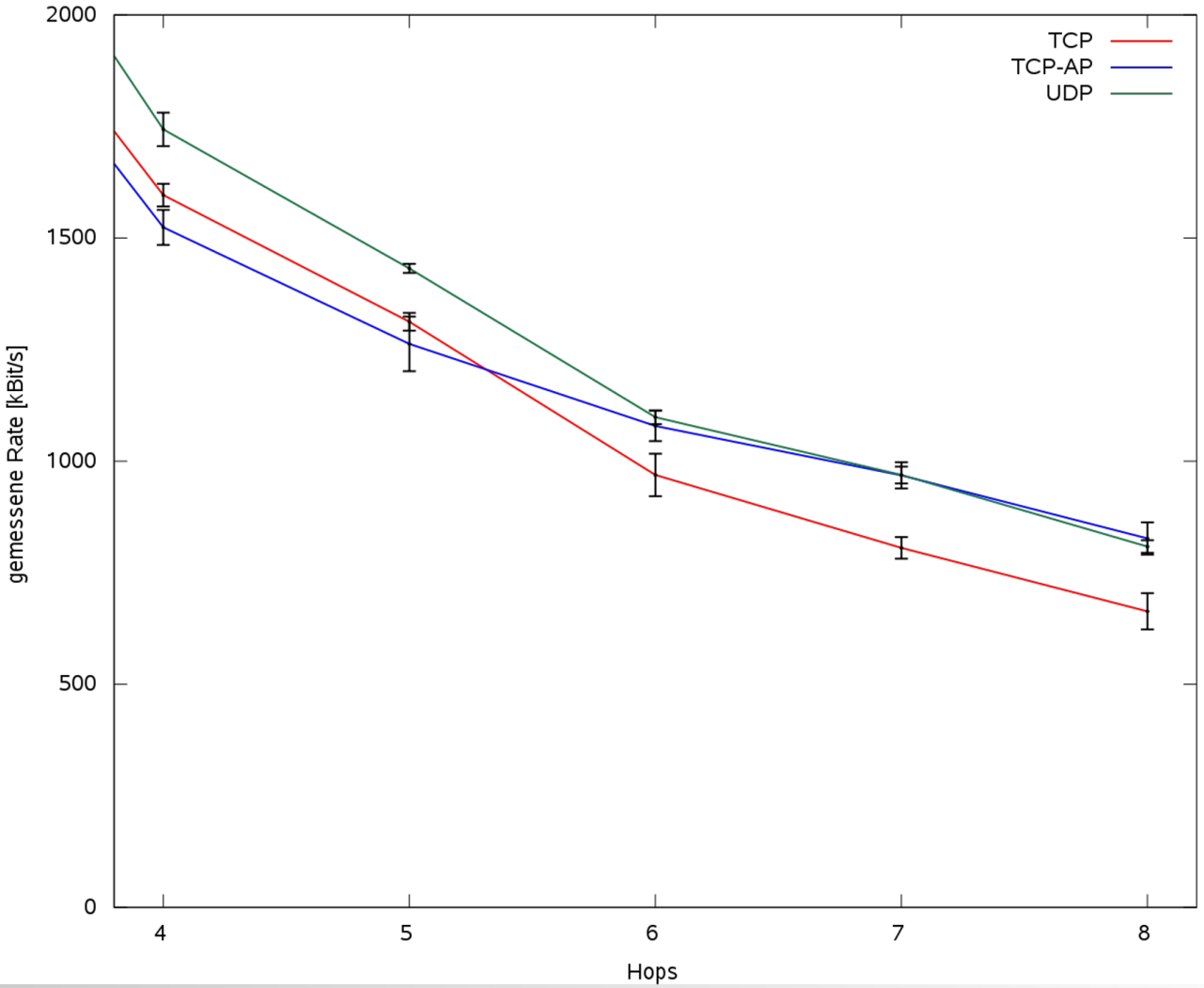


- Eine Übertragung über 1 bis 8 Hops
- Abstand entspricht Übertragungsreichweite
- Ziel: Vergleich des Datendurchsatzes von UDP, TCP und TCP-AP

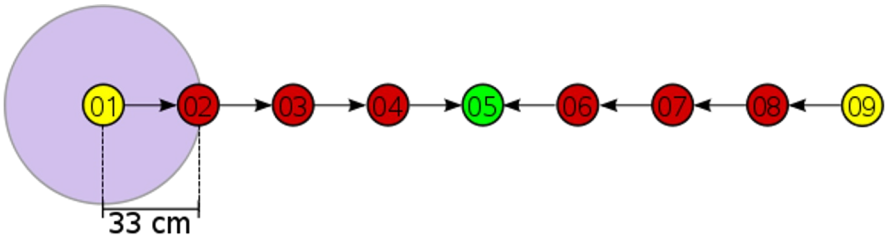
1. Experiment: Ergebnisse



1. Experiment: Ergebnisse

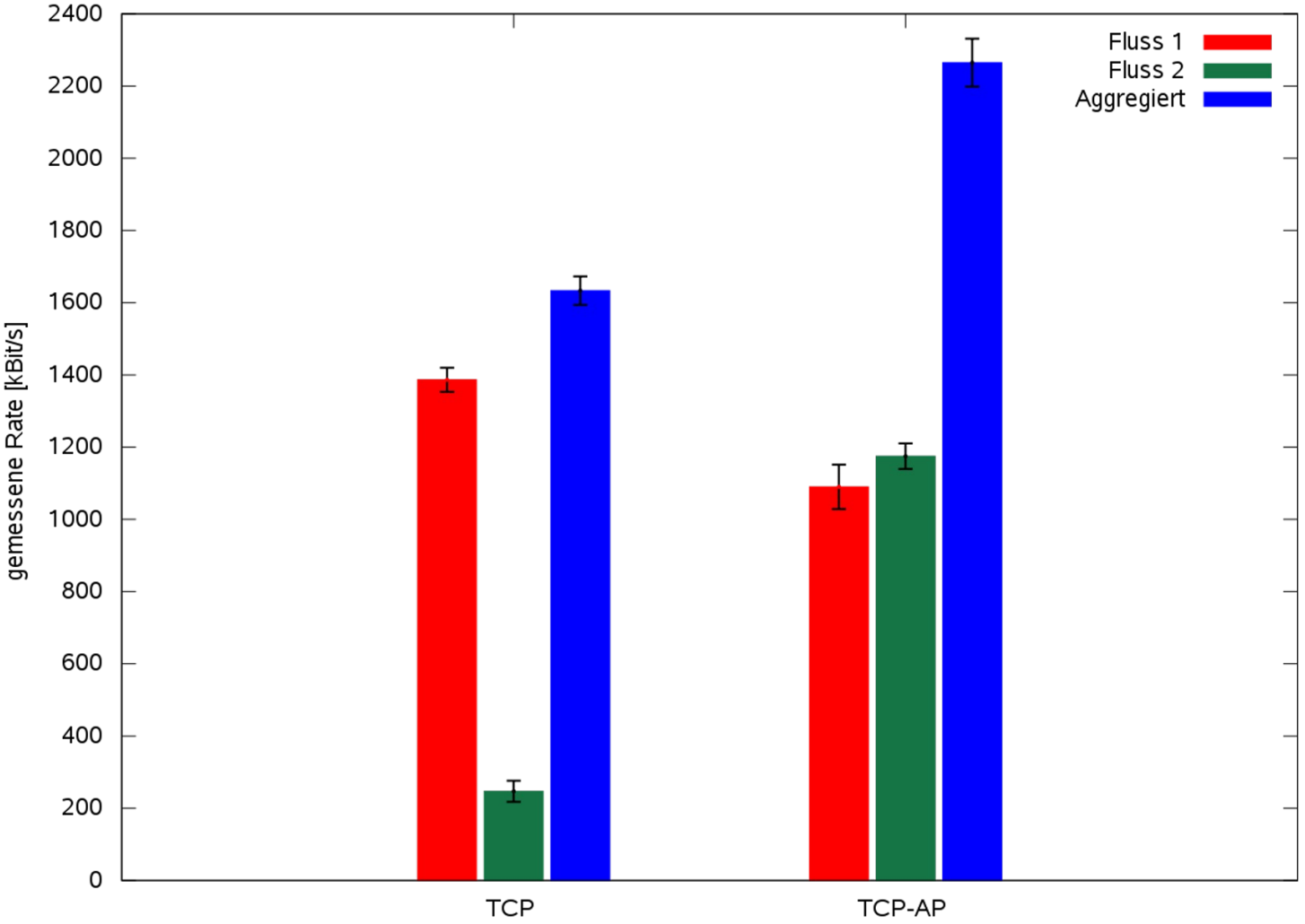


2. Experiment: Aufbau

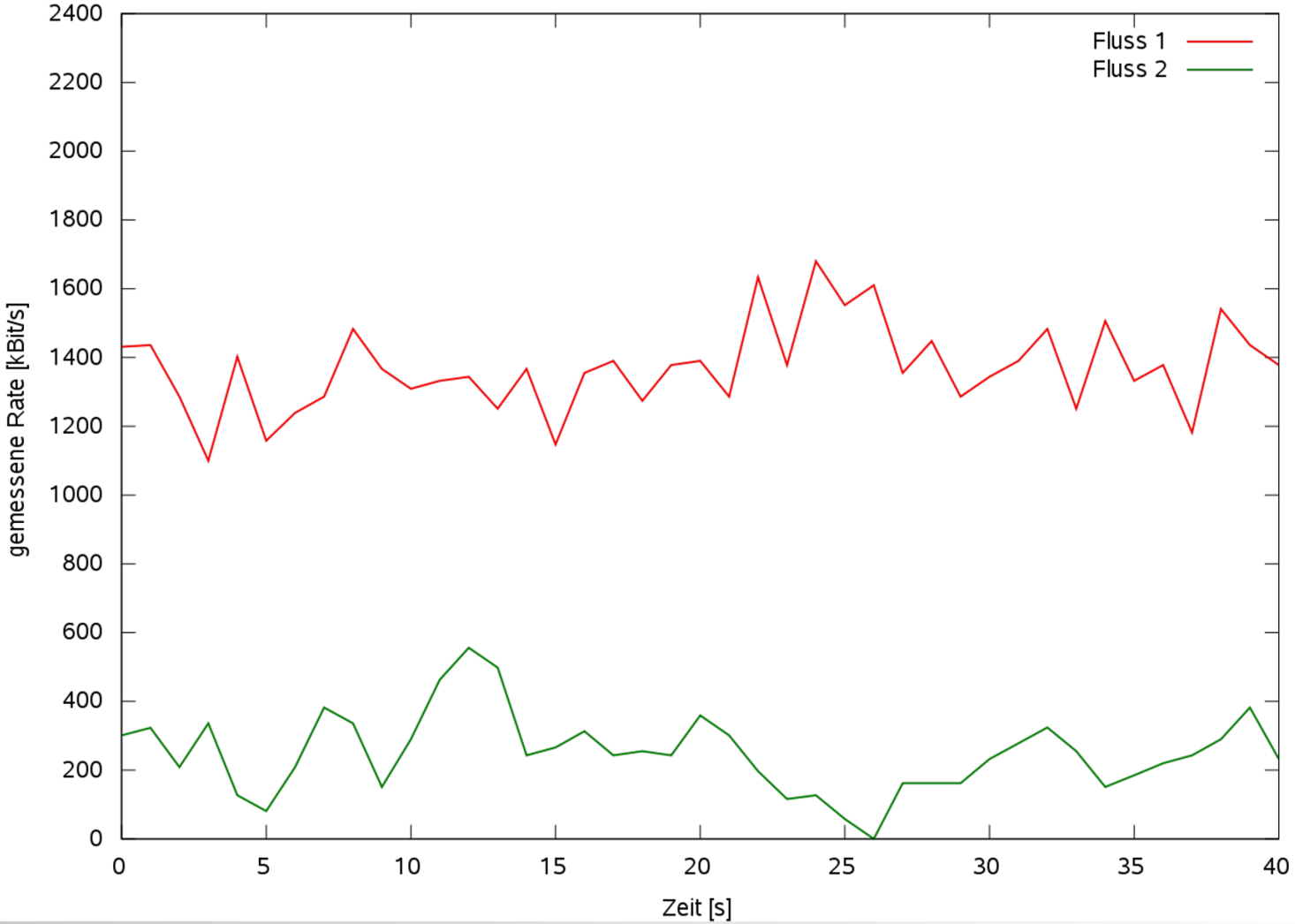


- Zwei zeitgleiche Übertragungen über je 4 Hops
- Abstand entspricht Übertragungsreichweite
- Ziel: Vergleich von Durchsatz und Fairness für TCP und TCP-AP

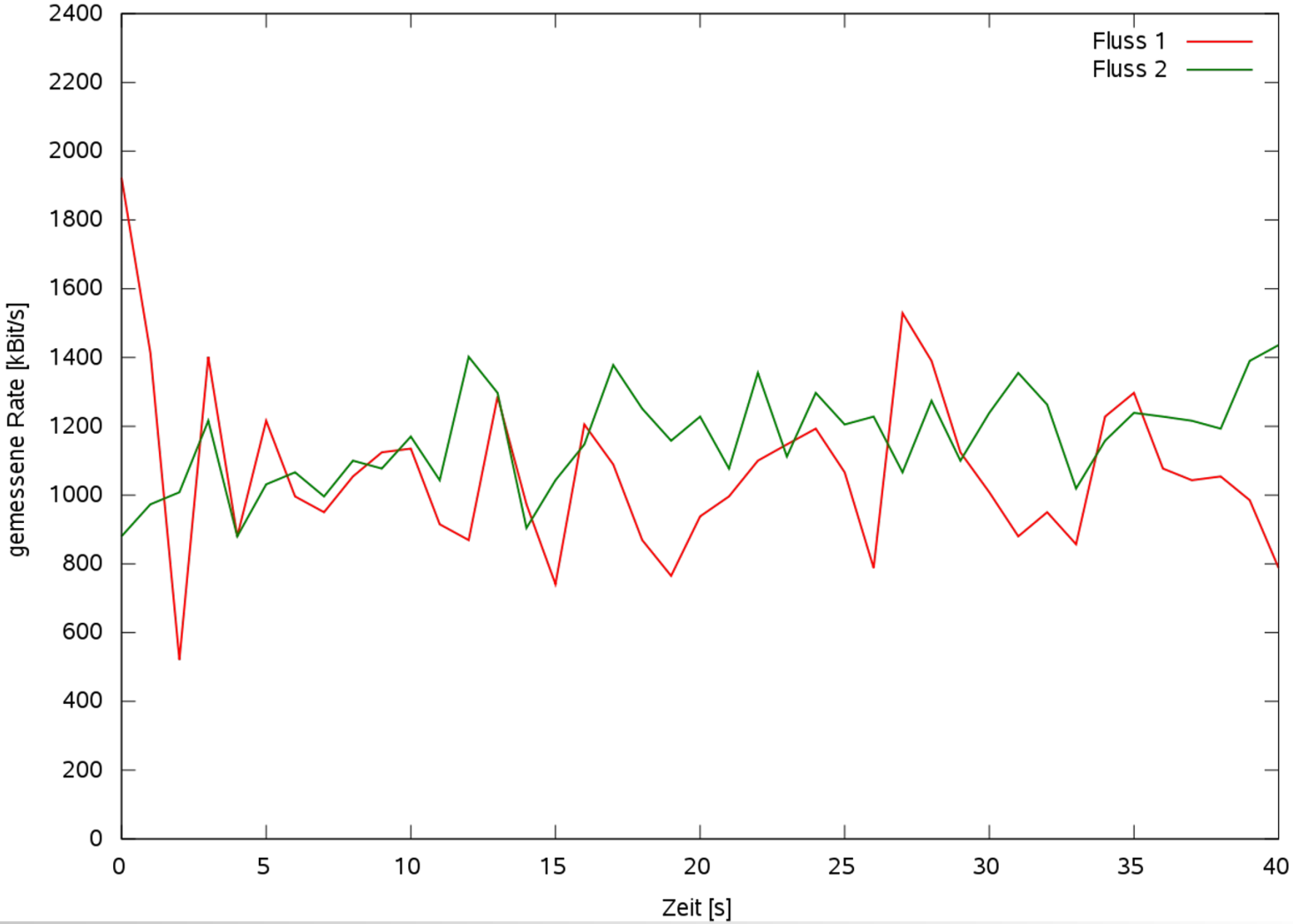
2. Experiment: Ergebnisse



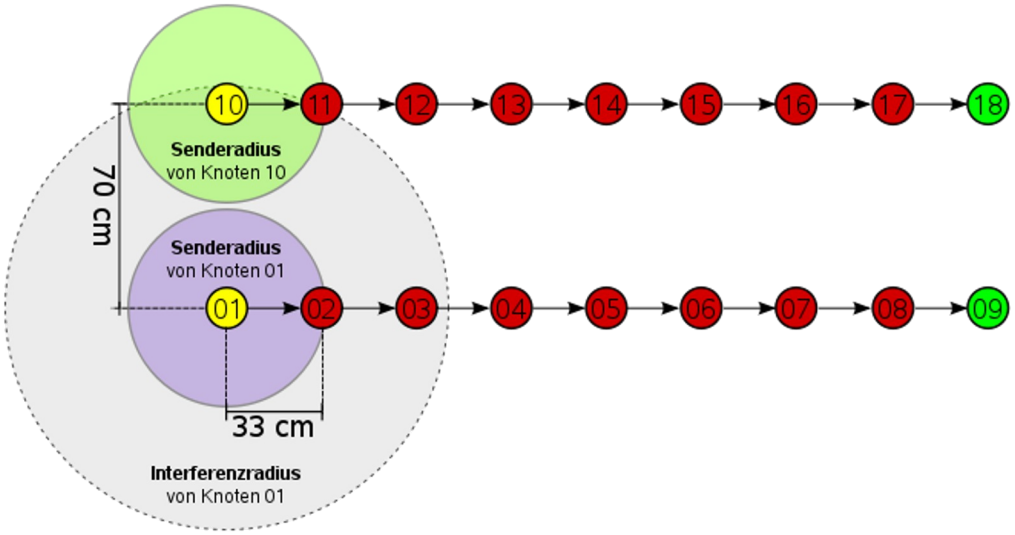
2. Experiment: TCP Übertragung



2. Experiment: TCP-AP Übertragung

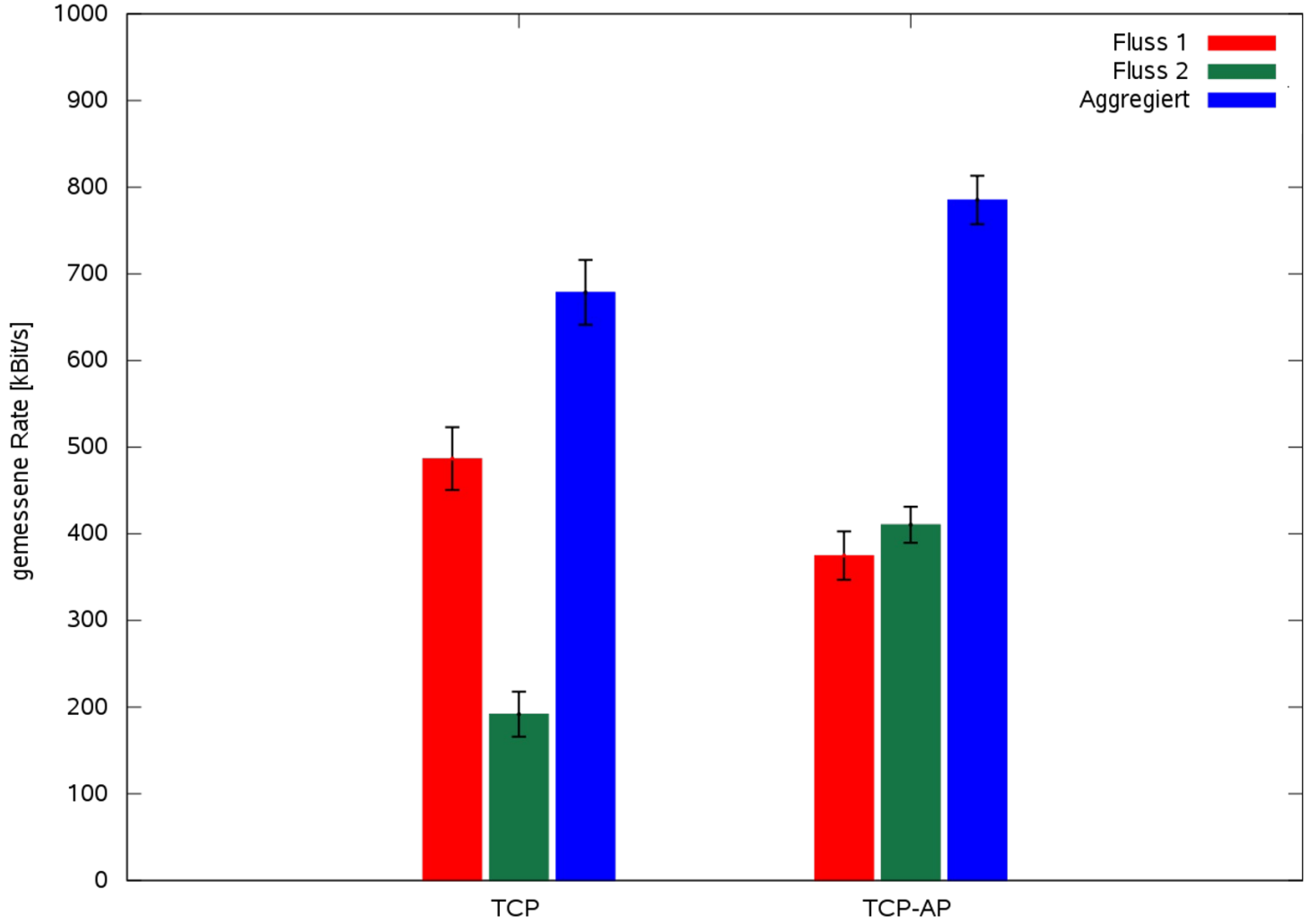


3. Experiment: Aufbau

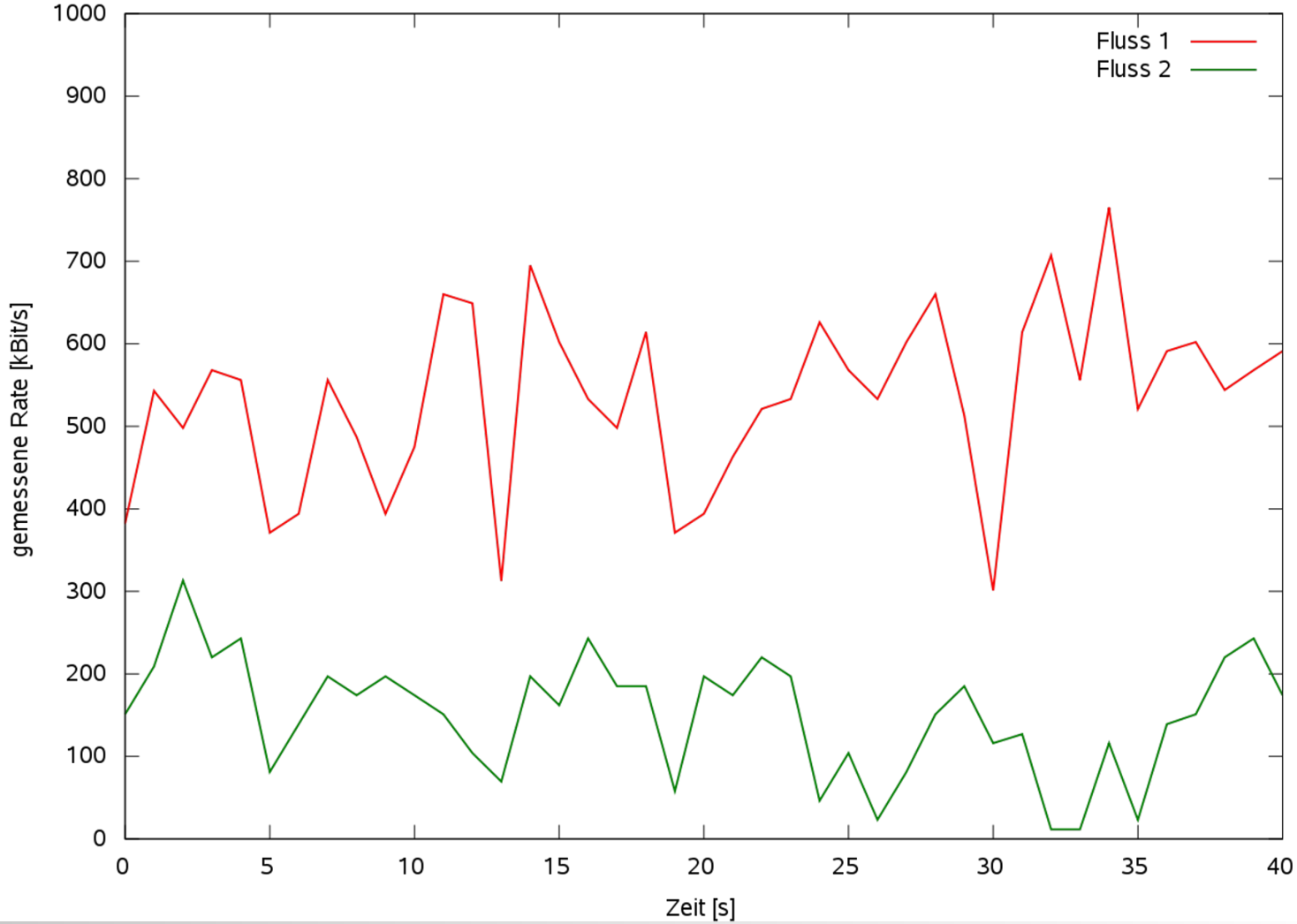


- Zwei zeitgleiche Übertragungen über je 8 Hops
- Abstand der Knoten entspricht Übertragungsreichweite, die Ketten befinden sich nur in Störreichweite
- Ziel: Vergleich von Durchsatz und Fairness für TCP und TCP-AP

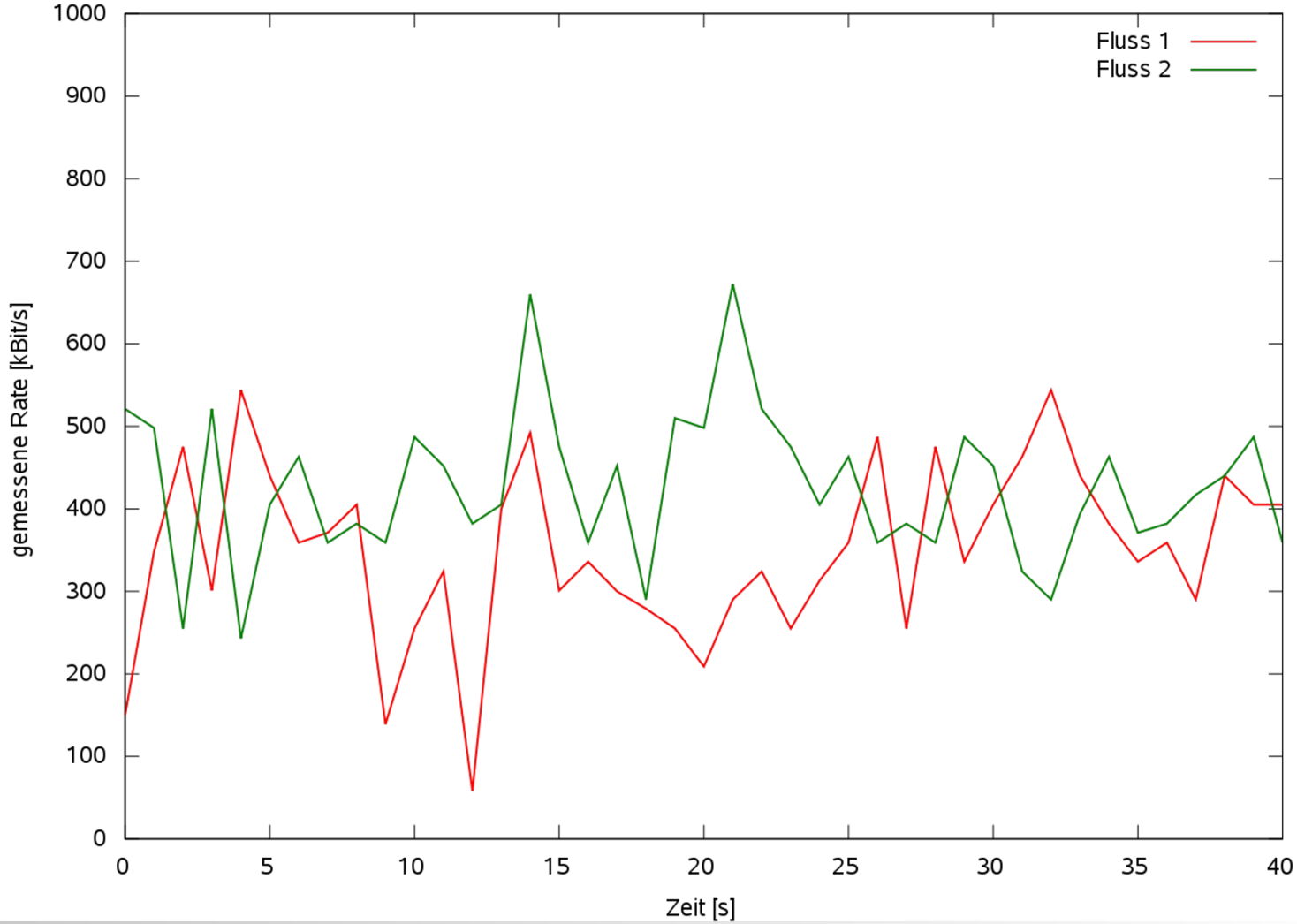
3. Experiment: Ergebnisse



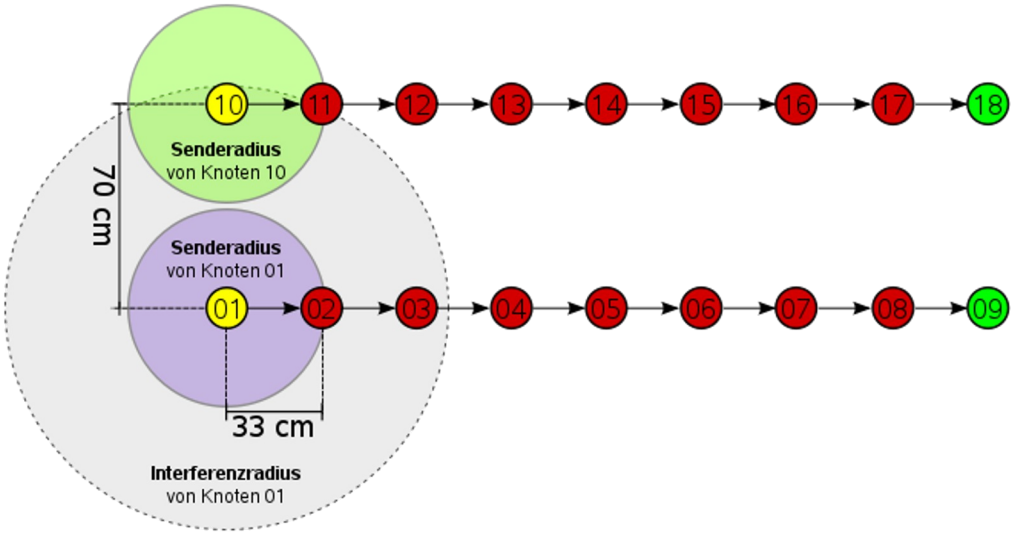
3. Experiment: TCP Übertragung



3. Experiment: TCP-AP Übertragung

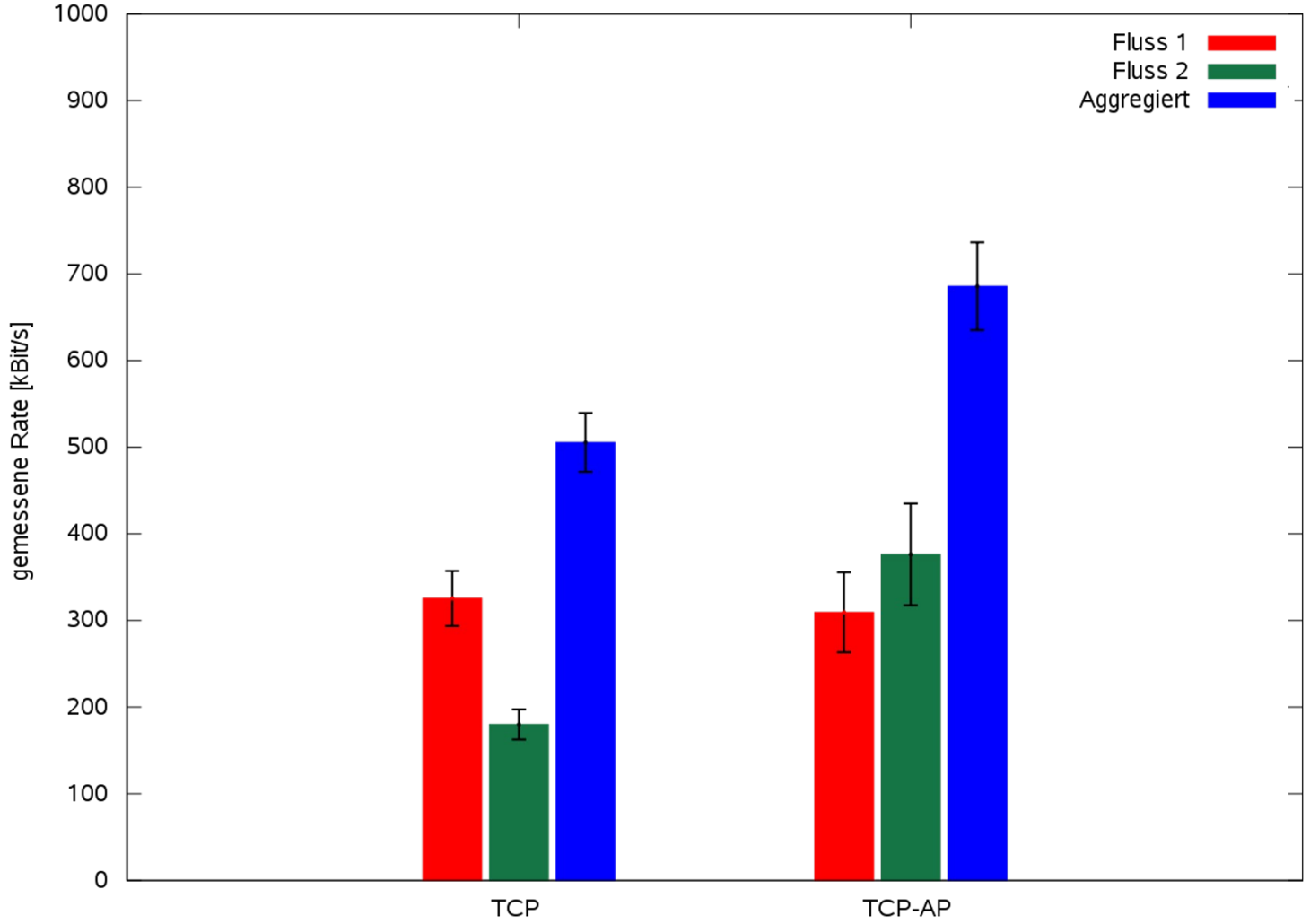


4. Experiment: Aufbau

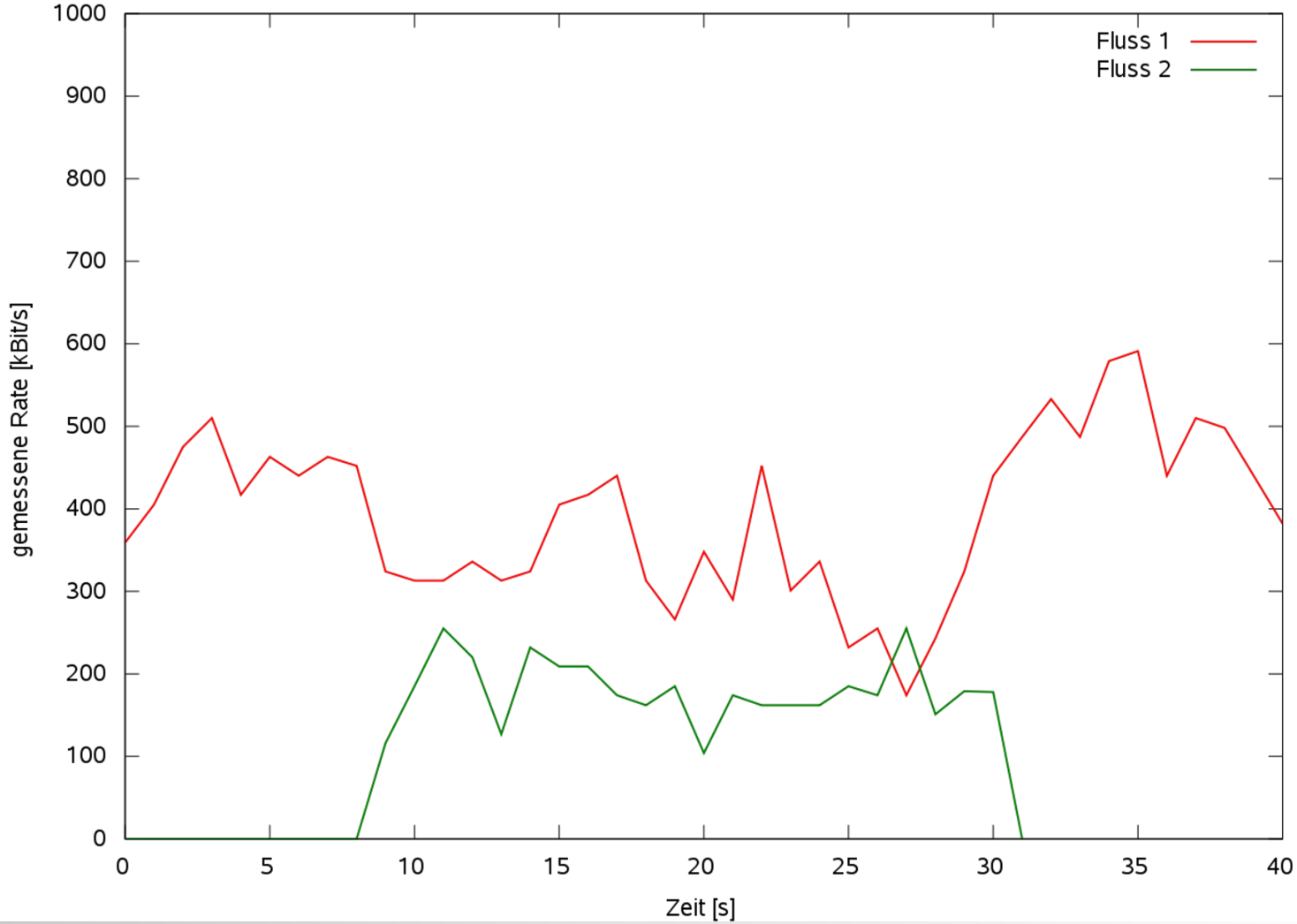


- Aufbau wie zuvor beim 3. Experiment
- Aber zwei asynchrone Übertragung:
  - 01 nach 09 von Anfang bis Ende des Experiments
  - 10 nach 18 startet 10s später und stoppt nach 20s
- Ziel: Vergleich von Durchsatz, Fairness und Reaktionszeit

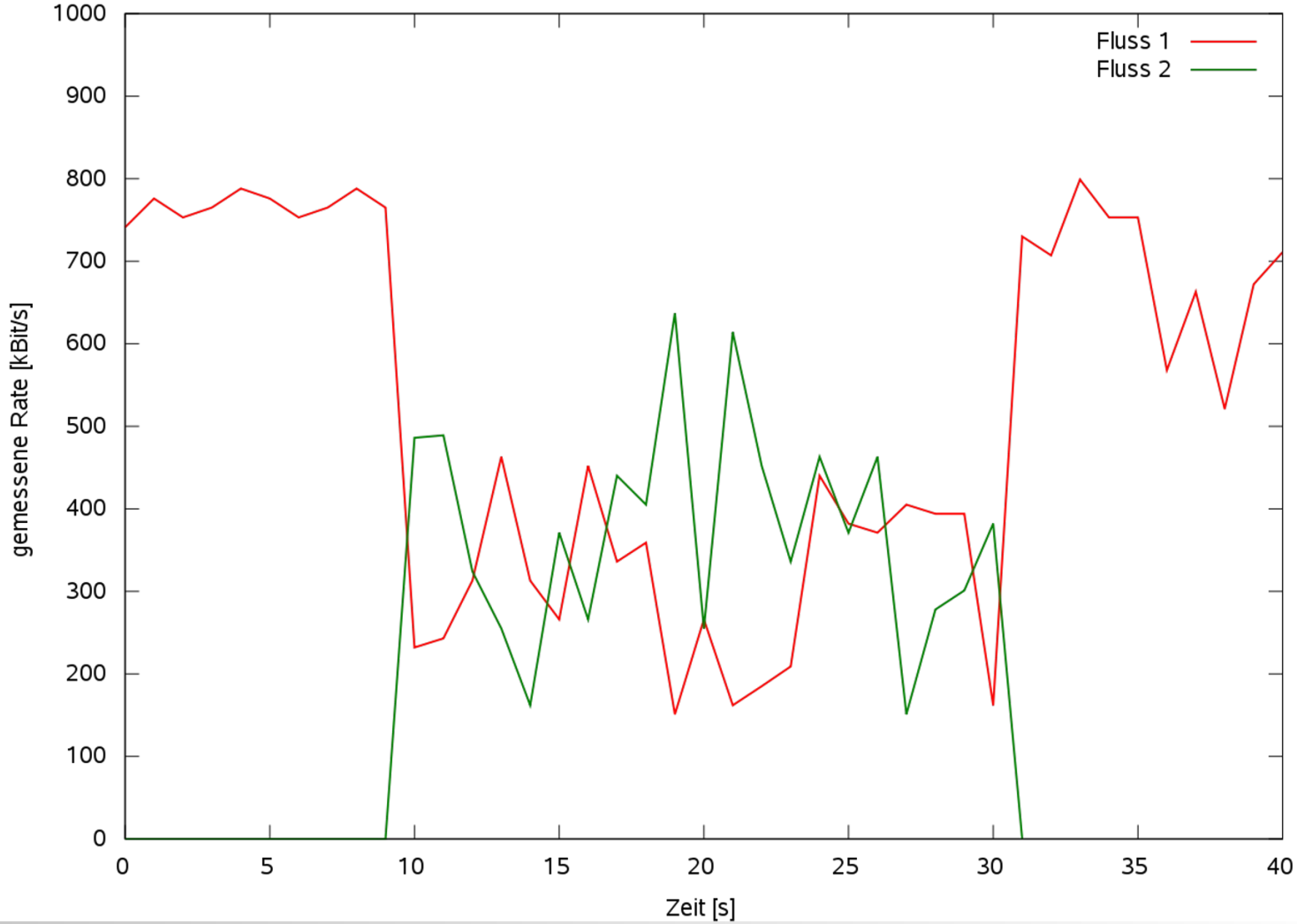
4. Experiment: Ergebnisse



3. Experiment: TCP Übertragung



4. Experiment: TCP-AP Übertragung



Zusammenfassung

.

Zusammenfassung

- Vorteile von TCP-AP:
  - TCP-kompatibel
  - Inkrementell einsetzbar
  - Berücksichtigung der Besonderheiten von Übertragungen in drahtlosen Netzen
- Verbesserung des Datendurchsatz auf multi-hop Pfaden in Mesh-Netzen
- Gute Fairness konkurrierenden Übertragungen in vielen Szenarien

Vielen Dank ...

Literatur

Quellenverzeichnis

Literaturverzeichnis

[1] ElRakabawy et al.: *TCP with adaptive pacing for multihop wireless networks.* MobiHoc 2005

[2] ElRakabawy et al.: *ScaleMesh: A Scalable Dual-Radio Wireless Mesh Testbed.* WiMesh 2008

[3] ElRakabawy et al.: *Practical Rate-Based Congestion Control in Wireless Mesh Networks.* KiVS 2009

[4] Nahm et al.: *TCP over multihop 802.11 networks: issues and performance enhancement.* MobiHoc 2005

[5] Saunders und Aragon: *Antennas and Propagation for Wireless Communication Systems.* Wiley&Sons, 2.Auflage, Mai 2007.

[6] David A. Rusling: *The Linux Kernel.* http://www.tldp.org/LDP/tlk/tlk.html

[7] Jain et al.: *A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems*. CoRR 1998.