Freie Universität Berlin

Der Retransmission Timeout von TCP

Philipp Lämmel

Proseminar Technische Informatik

Institut für Informatik, Betreuerin Dr. Katinka Wolter

- Während der Datenübertragung kommt TCP zum Einsatz
- Bei einer zu großen Netzwerkauslastung kann es zu Datenverlust kommen
  - Neusendung der Daten notwendig
- Auch bei Übertragungsfehlern ist erneute Übertragung des Paketes notwendig

➔ Wie entscheidet Protokoll, wann ein Paket neu geschickt werden soll?

Überblick

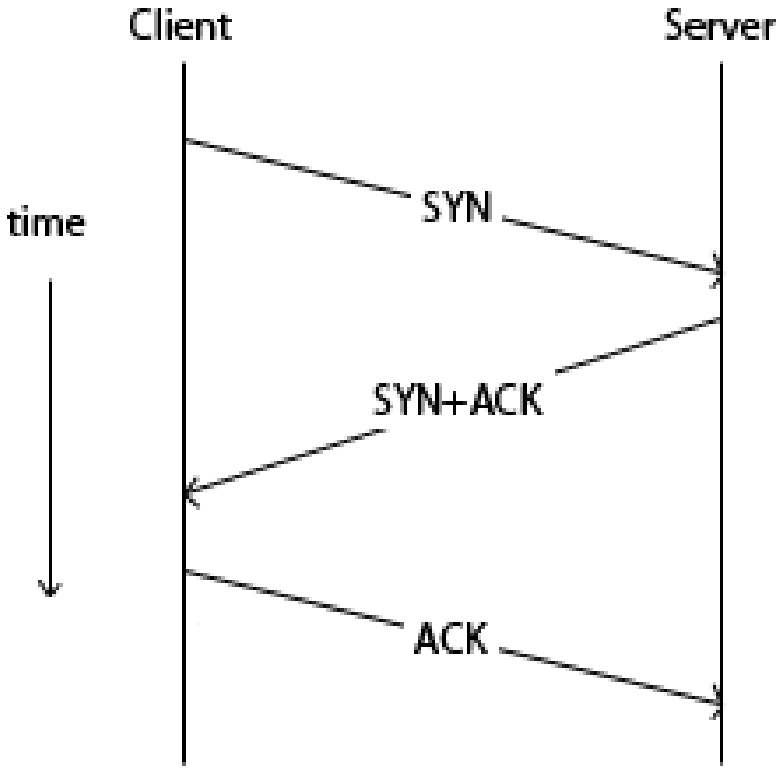
- Was ist das Transmission Control Protocol?
- Funktionsweise TCP
  - Verbindungsablauf
- Der Retransmission Timeout
  - Berechnung
  - Probleme des Timers
- Bewertung und Anwendbarkeit

Das Transmission Control Protocol

- Ab 1973 entwickelt
  - 1981 erste Standardisierung
- In der Transportschicht des OSI-Modells
- Für zuverlässige Übertragung zuständig
  - Gesichert durch Bestätigung der Pakete
- Übertragung in Form von Segmenten
  - Größe durch Maximum Transfer Unit bestimmt

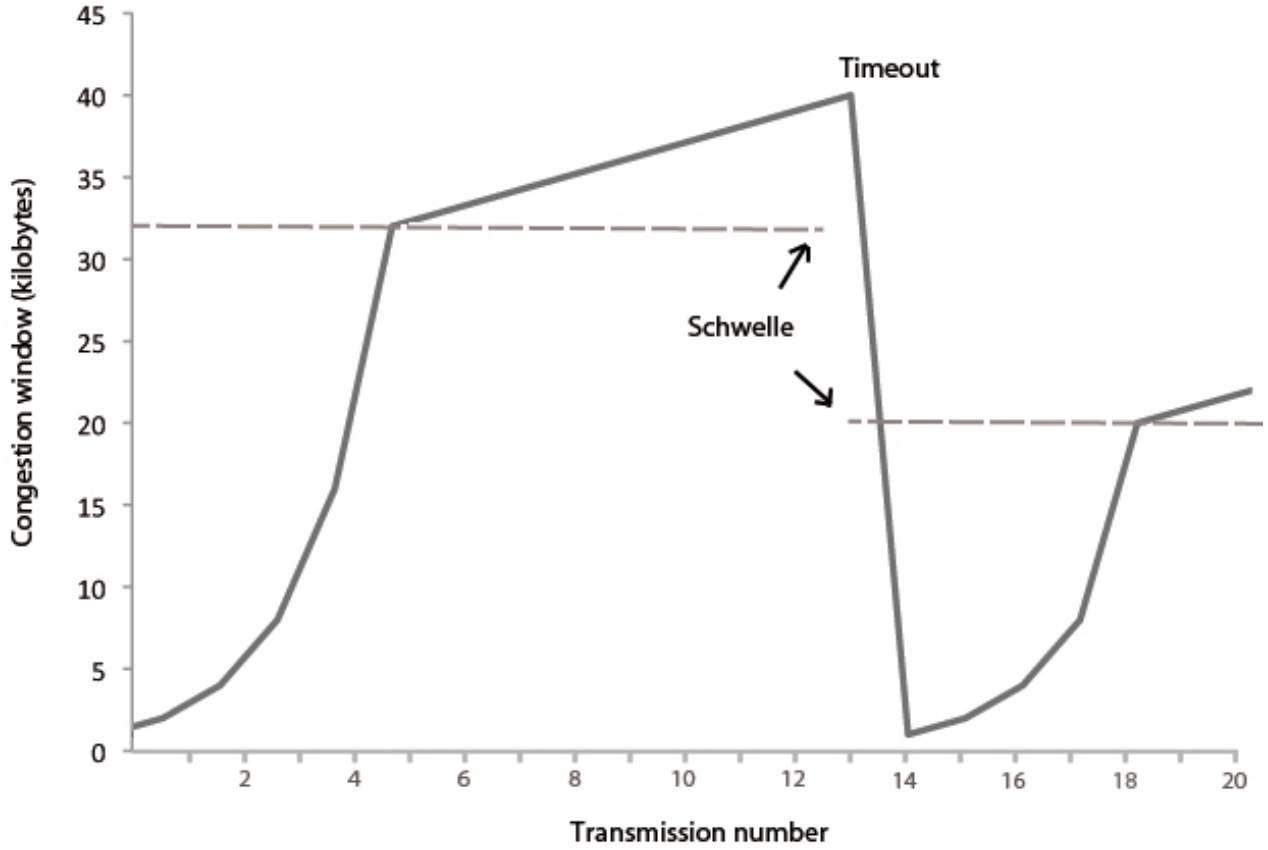
Verbindungsablauf TCP (1)

- Verbindung über Three-Way-Handshake hergestellt



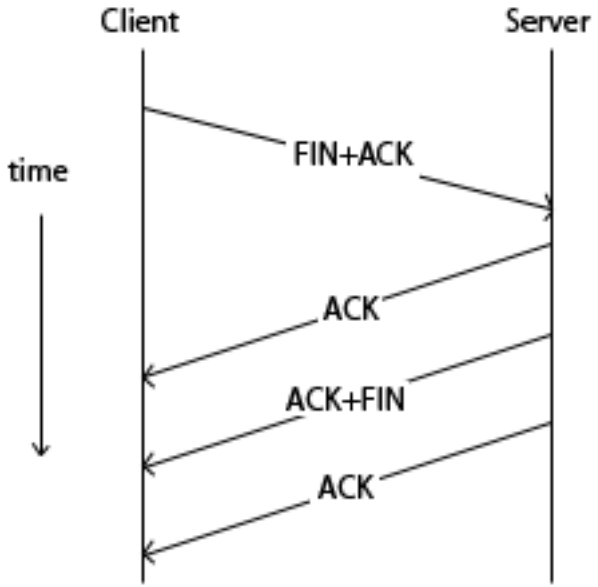
Verbindungsablauf TCP (2)

- Optimale Fenstergröße mittels slow-start Algorithmus bestimmt
  - ➔ Effizienzsteigerung
- Congestion-Avoidance Algorithmus
  - ➔ Anzahl an Timeouts soll verringert werden



Verbindungsablauf (3)

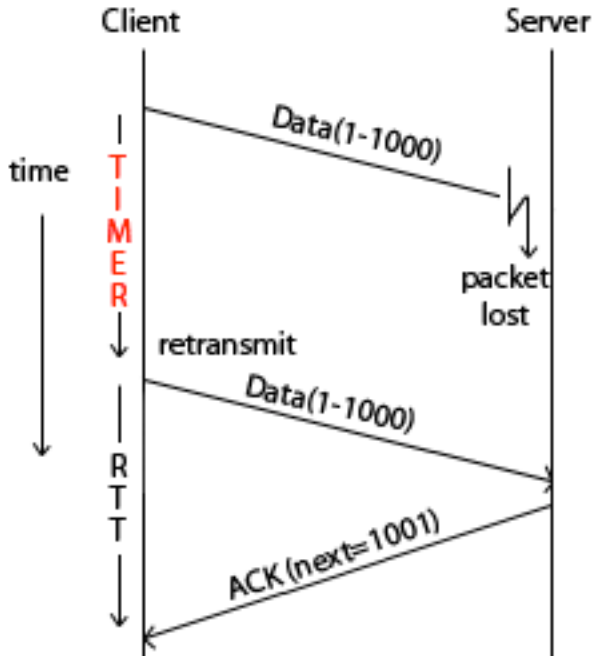
- Verbindungsabbau gegenseitig bestätigt



- Was passiert mit Segmenten, die bei der Übertragung verloren gingen? Wie entscheidet das Protokoll, wann ein Segment neu gesendet werden muss?

Der Retransmission Timeout

- Mittels Timer werden eventuelle Neusendungen bestimmt
  - Nach Ablauf der Zeit → Neusendung
- Der Timer wird dynamisch berechnet
- Berechnung in RFC 2988 festgelegt



Berechnung (1)

- Bei Initialisierung sollte Wert des Timers 2,5 - 3 sec betragen
- Nach ersten RTT Werte wie folgt angepasst:

$$
\begin{aligned}
SRTT &= R^1 && (1)\\
\sigma &= \frac{R^1}{2} && (2)\\
RTO &= SRTT + max(G, 4 \cdot \sigma) && (3)
\end{aligned}
$$

- Nach jedem ausgewerteten RTT R^i auf folgende Werte gesetzt:

$$
\begin{aligned}
\sigma &= (1-\beta) \cdot \sigma + \beta \cdot |SRTT - R^i| && (4)\\
SRTT &= (1-\alpha) \cdot SRTT + \alpha \cdot R^i && (5)
\end{aligned}
$$

- Es sollten $\alpha = 1/8$ und $\beta = 1/4$ sein

Berechnung (2)

- Timer muss Karn-Algorithmus benutzen:
  - Es werden nur RTT genommen, die ohne wiederholte Sendung bestätigt wurden

  ➔ Entgeht dem Problem, dass man sonst nicht bestimmen kann, welches bestätigt wurde
  - Zusätzlich RTO back-off verwenden

  ➔ Vor der erneuten Übertragung RTO um einen Faktor a erhöhen

Probleme des Timers (1)

- Zu Beginn der Übertragung keine Informationen über Beschaffenheit des Netzwerkes

→ Angemessener Initialwert für SRTT und damit RTO schwierig

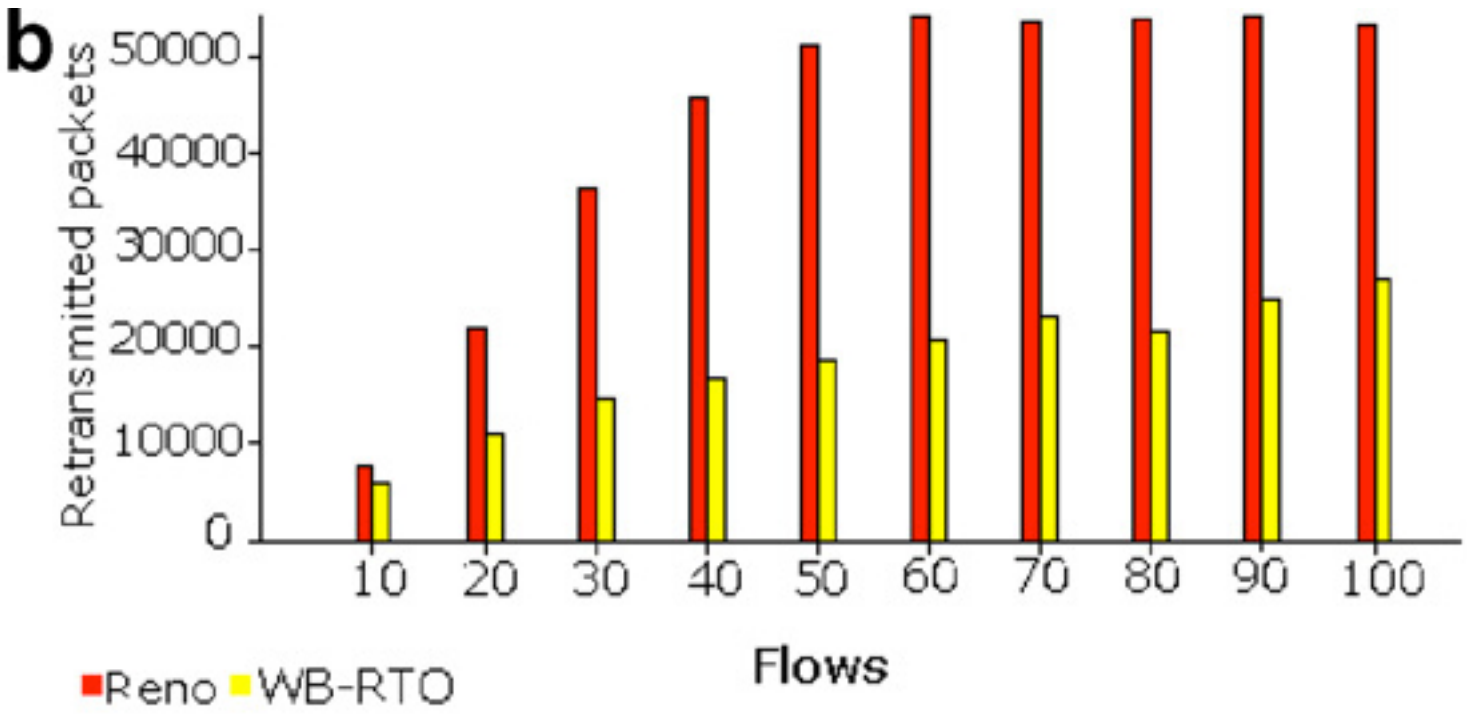
  - Wert zu lang → unnötiges Warten auf seine Datenübertragung
  - Wert zu kurz → überflüssiges Neusenden der Segmente

- Ursache für das Ablaufen des Timers nicht bestimmbar
  - TCP geht im Allgemeinen davon aus, dass es durch Überlast bedingt war

  → Algorithmen darauf angepasst

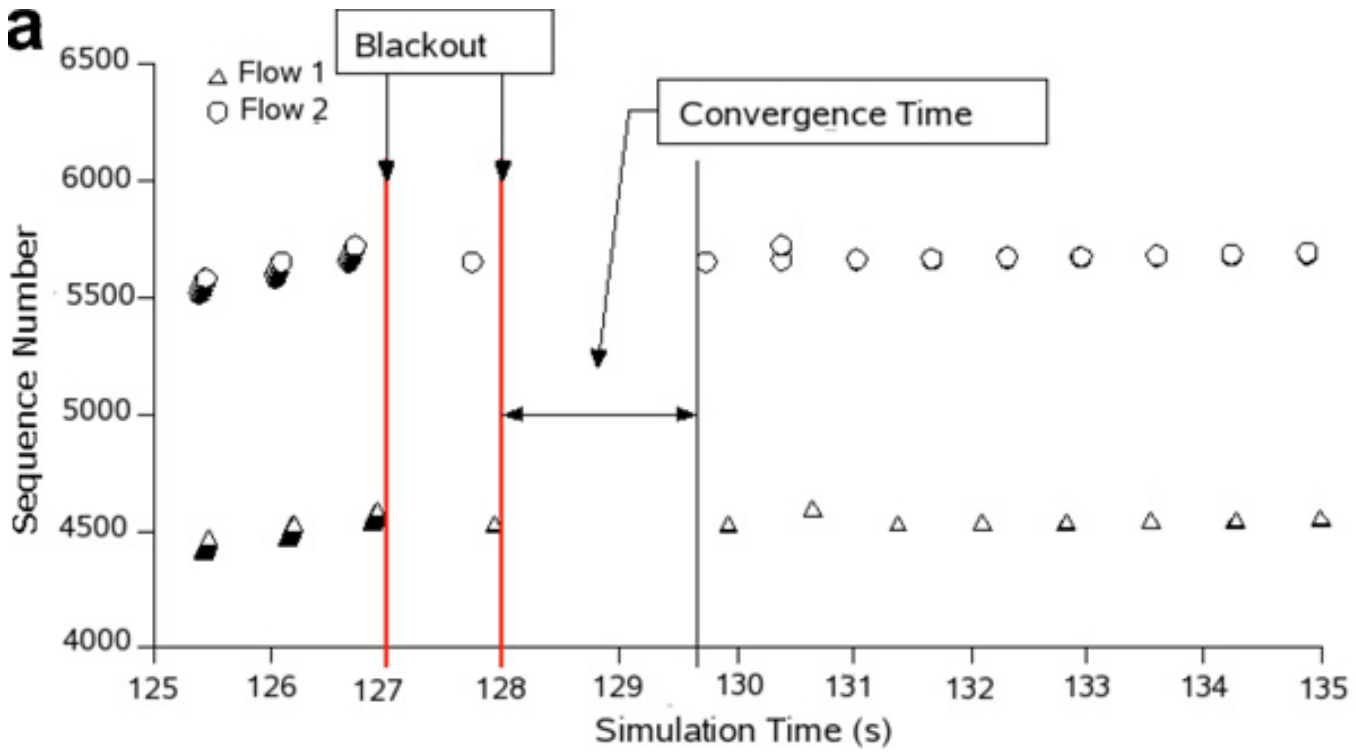
Probleme des Timers (2)

- Algorithmus beruht auf RTT Messungen
  - Spiegelt nur durchschnittliche Verzögerung des Netzwerkes wider
- Vernachlässigung von network contention in drahtlosen Netzwerken
  - Auswirkung auf Anzahl der erneuten Übertragungen



Probleme des Timers (3)

- In drahtlosen Netzwerken ist die Hauptursache nicht Überlast
  - Ausfall eines Knotens nicht unüblich
  - Fehlinterpretation des Protokolls → geht von Überlast aus



Bewertung und Anwendbarkeit (1)

- Trotz Probleme ist Algorithmus in drahtgebundenen Netzwerken zuverlässig und stabil
- Überlast bedingte Netzwerkveränderungen gut lösbar
  - Durch effiziente Anpassung des RTO-Wertes

- Jedoch ist Überlast nicht der Hauptgrund für Paketverlust in drahtlosen Netzwerken
  - Zum Beispiel durch Kollision oder dem Ausfall eines Knotens wesentlich wahrscheinlicher

  ➔ Schlechte Performance

➔ TCP ist nicht das Optimum in drahtlosen Netzwerken

Bewertung und Anwendbarkeit (2)

- Existieren Erweiterungen des Protokolls, die zwischen den Ursachen unterscheiden
  - Link Layer Protection Protocol (LHP) von X. Gao, S. N. Diggavi und S. Muthukrishnan

- Durch Explicit Loss Notification besteht Möglichkeit zur Unterscheidung zwischen Ursachen
  - Realisiert durch das Schützen des Link Layer Header

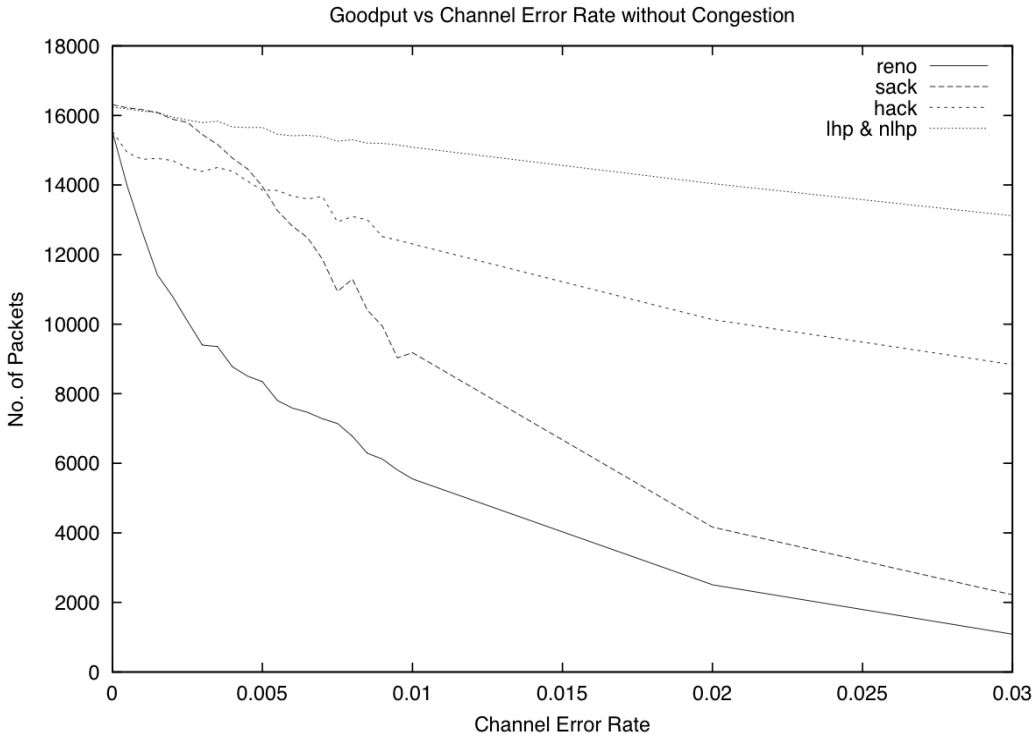
  ➔ Kommt dieser an, auch wenn IP Header korrupt ist, gab es einen link failure
  - Mit speziellen ACK weiß Sender, dass es link failure war

  ➔ Muss nicht auf Congestion eingegangen werden

Bewertung und Anwendbarkeit (3)

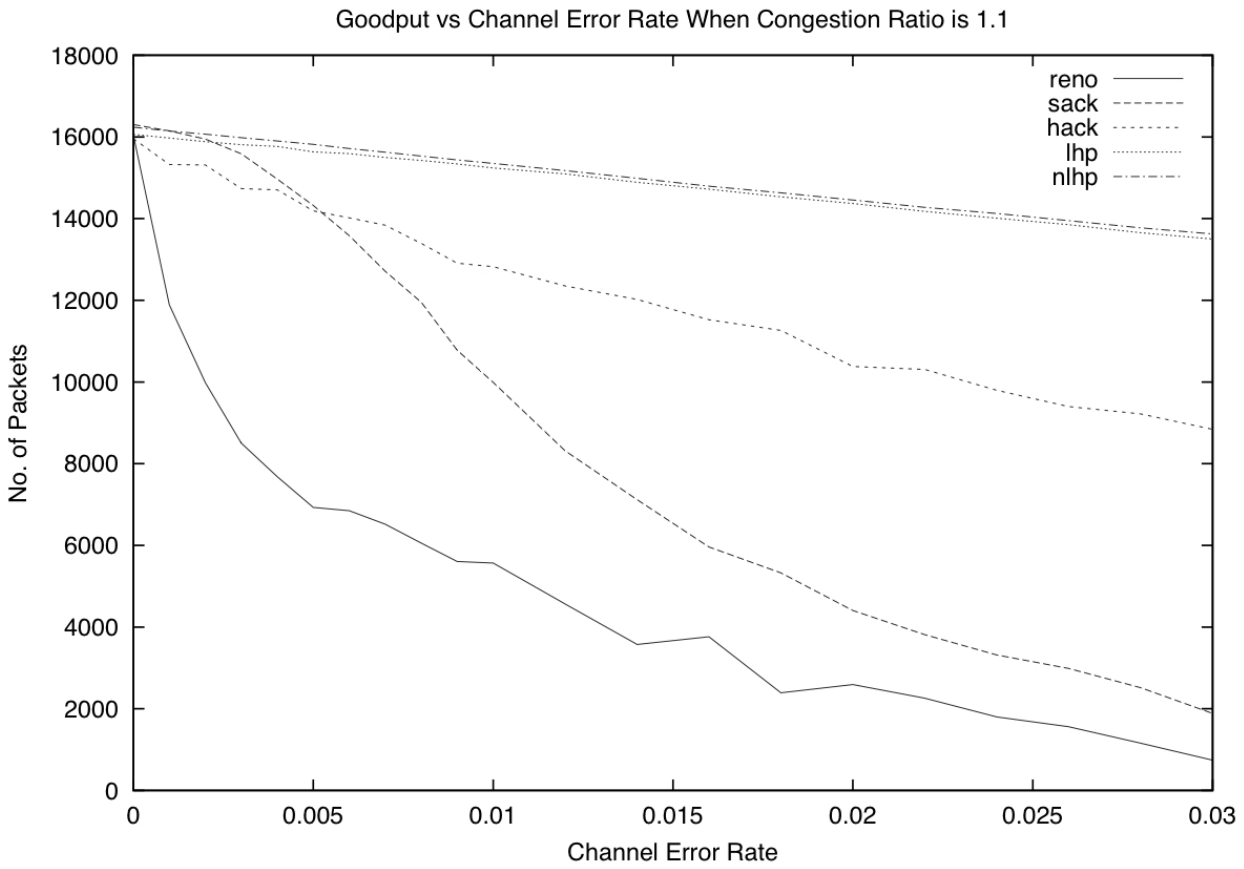
- LHP übertragt per stop-and-wait
  - ➔Pakete können nicht in falscher Reihenfolge ankommen

- Bewertung LHP gegenüber TCP (ohne Congestion):



Bewertung und Anwendbarkeit (4)

- Bewertung LHP gegenüber TCP (mit Congestion):



Fazit

- Existieren Erweiterungen, die in drahtlosen Netzwerken besser geeignet sind
  - Zum Beispiel LHP
- Erweiterungen besitzen Möglichkeit zwischen den Ursachen für das Ablaufen des Timers zu unterscheiden
  - Hauptgrund für schlechte Performance in drahtlosen Netzwerken

Vielen Dank für Ihre Aufmerksamkeit!

Literatur

[01] X. Gao, S. N. Diggavi, and S. Muthukrishnan. LHP: An end-to-end reliable transport protocol over wireless data networks

[02] V. Jacobson. Congestion avoidance and control

[03] R. E. Kahn and V. G. Cerf. Transmission Control Protocol - Darpa Internet Program Protocol Speficiation

[04] P. Karn and C. Partridge. Improving Round-Trip Time Estimates in reliable transport protocols

[05] V. Paxson and M. Allman. Computing TCP's Retransmissio Timer

[06] I. Psaras and V. Tsaoussidis. Why TCP timers (still) don't work well

[07] F. Stann and J. Heidemann. RMST: Reliable Data Transport in Sensor Networks

[08] A. S. Tanenbaum. Computer Networks

[09] K. Washburn and J. Evans. TCP/IP Aufbau und Betrieb eines TCP/IP-Netzes

[10] L. Zhang. Why TCP timers don't work well