



Rechnernetze I, WS 2003/2004

TCP-Verstopfungskontrolle (Ergänzung zum Vorlesungsskript)

Andreas Schorr



QoS Group

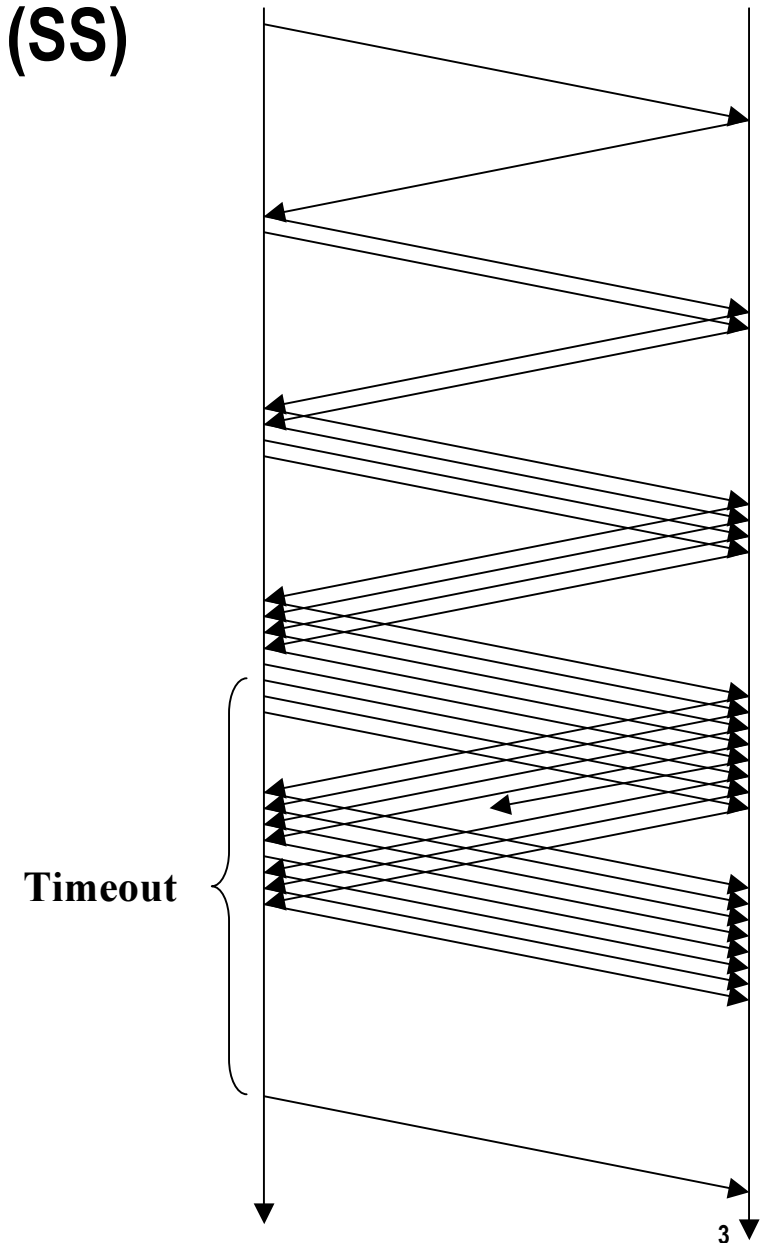
TCP-Verstopfungskontrolle

- Flusskontrolle allein reicht aus für Punkt-zu-Punkt-Verbindungen.
- Wenn in einem Netz mit Zwischenknoten (Routern) viele Sender gleichzeitig aktiv sind, kann es jedoch auf den Routern zu Verstopfung kommen, selbst wenn alle Endknoten Flusskontrolle auf der Transportschicht machen.
- TCP verwendet ein Fenster (*Advertised Window*) zur bereits bekannten Flusskontrolle (zwischen den Endknoten), und zusätzlich ein sogenanntes Verstopfungsfenster (*Congestion Window*). Sender darf nur soviel schicken, wie in das kleinere dieser beiden Fenster hineinpasst.
- Der Empfänger bestimmt beim Verbindungsaufbau die Größe *awnd* des *Advertised Window*.
- Der Sender bestimmt die Größe *cwnd* des *Congestion Window* entsprechend der aktuellen Verstopfungssituation im Netz.
- Im Falle einer Verstopfung muss der gesamte Datenverkehr gedrosselt werden. Dazu kommen bei TCP die Mechanismen „*Slow Start*“ (SS) und „*Congestion Avoidance*“ (CA) zum Einsatz.

Slow Start (SS)

- *cwnd* wird mit 1 initialisiert (1 Segment = 512 Byte)
- Bei Erhalt eines ACK wird *cwnd* um 1 erhöht.
- Exponentielles Wachstum von *cwnd* (nach der ersten RTT um 1, dann um 2, dann um 4, dann um 8, ...)
- Falls Timeout abläuft → SS startet neu (*cwnd* = 1)

*Hinweis: TCP erlaubt, dass der Empfänger entweder jedes oder jedes zweite Datenpaket mit einem ACK bestätigt. Daher kann das Wachstum von *cwnd* je nach Verhalten des Empfängers etwas variieren.*



Congestion Avoidance (CA)

- Exponentielles Wachstum von *cwnd* nur solange $cwnd \leq sstresh$ (*Slow-start threshold*).
 - Lineares Wachstum von *cwnd* um 1 pro RTT, sobald $cwnd > sstresh$ (*CA-Phase*).
 - Am Anfang einer TCP-Verbindung wird *ssresh* auf 128 Segmente (=65536 Bytes) gesetzt.
 - Kommt es zu einem Timeout, dann wird *ssresh* auf die halbe aktuelle Größe des Sendefensters gesetzt – mindestens jedoch auf eine Größe von 2 Segmenten → $ssresh = \max(2, 0,5 * \min(cwnd, awnd))$.
- Nach einem Timeout geht der Algorithmus bereits dann in die CA-Phase, wenn *cwnd* halb so groß ist wie zu dem Zeitpunkt, als es beim letzten Mal Probleme gegeben hat.

Hinweis: Die Beschreibung der SS- und CA-Algorithmen entspricht deren Definition in RFC 2001. Die aktuellste (etwas veränderte) Definition ist in RFC 2581 zu finden.

Fast Retransmit

- Ein ACK enthält immer die Sequenznummer des nächsten erwarteten Segments (bei Ankunft von Paket $n-1$ also z.B. die Nummer n).
- Hat ein Empfänger das Segment n nicht erhalten, dann schickt er bei Ankunft von $n+1$, $n+2$, usw. immer ein ACK mit der Nummer n . Diese ACKs nennt man *Duplicated ACKs*, da sie alle dieselbe Nummer wie das ACK enthalten, das bei Ankunft von Segment $n-1$ verschickt wurde.
- *Duplicated ACKs* können entstehen, wenn
 - sich Segmente im Netz überholen,
 - Segmente verloren gehen.
- Selbst im Internet ist es unwahrscheinlich, dass ein Paket von mehreren anderen Paketen überholt wird. Daher werden 3 aufeinanderfolgende *Duplicated ACKs* als Zeichen für einen Paketverlust gewertet.
- Anstatt den Timeout abzuwarten, reagiert TCP, indem es das unbestätigte Paket sofort neu überträgt (*Fast Retransmit*).

Fast Recovery

- 3 aufeinanderfolgende *Duplicated ACKs* werden als Zeichen für eine Verstopfung im Netz gewertet. Die Verstopfungskontrolle kann also sofort etwas tun, anstatt den Timeout abzuwarten.
 - Durch die *Duplicated ACKs* erhält der Empfänger jedoch die Information, dass offensichtlich noch Pakete beim Empfänger ankommen. Die Verstopfung ist also nicht „ganz so schlimm“, als wenn gar nichts mehr ankommt (d.h. wenn der Timeout zuschlägt).
 - Genau wie bei einem Timeout wird *sstresh* auf $\max(2, 0,5 * \min(cwnd, awnd))$ gesetzt. *cwnd* wird jedoch nicht auf 1, sondern ebenfalls auf *sstresh* gesetzt.
- Wie bei einem Timeout geht der Algorithmus nun also in die CA-Phase, wenn *cwnd* bereits halb so groß ist wie zu dem Zeitpunkt, als ein Problem aufgetaucht ist. Anstelle von SS beginnt jedoch sofort die CA-Phase.

Hinweis: Der als „Fast Recovery“ bekannte Mechanismus ist in Wirklichkeit noch etwas komplizierter (siehe RFC 2581).

Einschränkungen

- **Slow Start, Congestion Avoidance, Fast Retransmit und Fast Recovery gehen von der Annahme aus, dass der Verlust von Daten immer gleichbedeutend ist mit Verstopfung im Netz.**
 - **In den meisten kabelgebundenen Netzen ist die Annahme auch fast immer richtig.**
 - **In störanfälligen (drahtlosen) Netzen kann man im Allgemeinen nicht von der Richtigkeit dieser Annahme ausgehen. Die TCP-Verstopfungskontrolle führt in derartigen Netzen zu unnötiger Durchsatzverminderung im Falle von Paketverlusten als Folge von Störungen.**
- **Zusätzliche Mechanismen müssen das Eingreifen der TCP-Verstopfungskontrolle unterbinden.**