TCP

TCP

Paketverluste entstehen aufgrund von

- Überlast, defekten Routern, nicht ausreichender Puffer-Kapazität des Empfängers oder Bitfehlern während der Übertragung.
- Weitere Fehlerquellen sind duplizierte Pakete sowie das Eintreffen von Paketen in falscher Reihenfolge.

TCP, das **Transmission Control Protokoll**,
ist die verantwortliche Kontrollinstanz für die Behebung von Überlast und die bestmögliche Nutzung der Link-Kapazitäten.

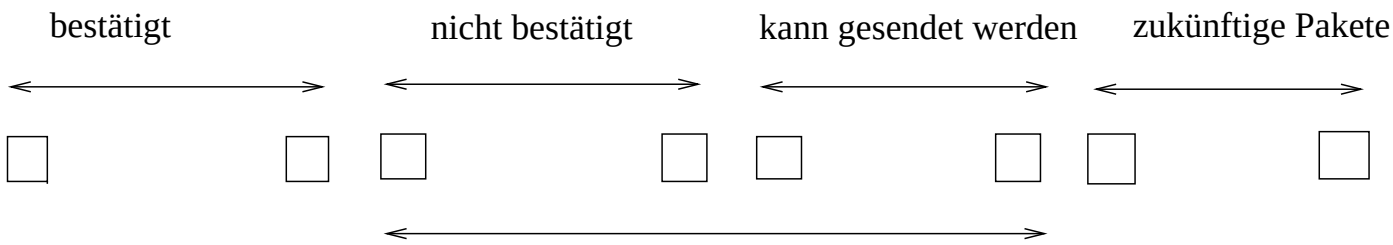
- Die TCP-Verkehrskontrolle misst dazu die Belastung einer Sender-Empfänger Verbindung und stellt den bisher beobachteten Durchsatz der Verbindung fest.
- Die Emissionsrate wird adaptiv auf die Belastung durch Hoch- oder Runterfahren eingestellt.

Sendefester, Acks, RTT und RTO

Das Sendefenster

Der Sender berechnet ein Sendefenster (**sliding window**), dessen Größe sich dynamisch dem Durchsatz anpasst, aber die Länge des Empfänger-Puffers nicht übersteigen sollte.

- Das Sendefenster besteht aus einem Intervall aufeinanderfolgender Pakete und beschreibt die versandbereiten Pakete, bzw. die versandten, aber noch nicht bestätigten Pakete.
  - Pakete werden durchnummeriert: Der Empfänger kann die Pakete in richtiger Reihenfolge zusammenzusetzen.
  - Der Empfänger bestätigt den Erhalt eines Pakets durch **Acknowledgements** oder **Acks**, die Paketen in der Gegenrichtung hinzugefügt werden.



Sliding Window (Länge = Länge des Empfänger Puffers)

Ack's

- Eine **kumulative Bestätigung** bestätigt stets auch den Erhalt aller nach der letzten kumulativen Bestätigung gesandten Pakete.
- Eine **duplizierte Bestätigung** wird verschickt, wenn ein Paket in falscher Reihenfolge empfangen wird:
  - Das jüngste in richtiger Reihenfolge empfangene Paket wird wiederholt bestätigt.
  - Eine duplizierte Bestätigung ist für den Sender ein Hinweis, aber kein Beweis, für einen Paketverlust.
- Eine **selektive Bestätigung** führt bis zu vier Paket-Intervalle explizit auf, die erfolgreich empfangen wurden.
  - Selektive Bestätigungen sind wichtig, wenn mehrere Paketverluste in einem Sendefenster auftreten.
  - Sie helfen, die nachzusendenden Pakete zu identifizieren.

Drohender Paketverlust wird frühzeitig erkannt und wiederholt zu verschickende Pakete können bestimmt werden.

Rundreisezeit und Retransmission Timeout

- Der Sender berechnet die Rundreisezeit (bzw. **Round-Trip Time** oder **RTT**) eines Pakets.
  - Die Rundreisezeit beginnt mit dem Versand und endet mit dem Erhalt des vom Empfänger versandten Ack.
  - Die Rundreisezeit wird durch Stichproben aktualisiert.
- Der **Retransmission Timeout** (**RTO**) ist der Zeitraum, in dem der Sender auf eine Bestätigung eines versandten Pakets wartet.
  - Geht innerhalb des RTO-Zeitraums keine Bestätigung ein, wird das Paket wieder gesendet.
- Der RTO-Zeitraum verändert sich dynamisch mit der Rundreisezeit:
  - Bei steigender Rundreisezeit, und damit bei stärker belasteten Links, wird RTO entsprechend hochgesetzt und der Sender wartet länger auf eine Bestätigung.

Slow Start, Congestion Avoidance, Fast Retransmit und Fast Recovery

Anpassung des Sendefensters

Der Sender versucht, sein Sendefenster den Netzbedingungen anzupassen und benutzt dazu die Komponenten

- **Slow Start**,
- **Congestion Avoidance**,
- **Fast Retransmit** und
- **Fast Recovery**.

- Die TCP-Varianten TCP Tahoe, TCP Vegas, TCP Reno, TCP New Reno, TCP Sack unterschieden sich zum Teil wesentlich in der Implementierung dieser vier Komponenten.
- Wir beschreiben TCP Sack.

Slow Start und Congestion Avoidance

Slow Start wird beim Aufbau einer Verbindung bzw. nach einem RTO-Fehler aufgerufen.

- **Slow Start** beginnt mit einem kleinen Sendefenster S_0 und endet, wenn der Schwellenwert S_1 ($S_0 << S_1$) überschritten wird.
- Werden alle bisher versandten Pakete innerhalb des Retransmission Timeouts bestätigt, dann setze die Länge des gegenwärtigen Sendefensters S auf $S = \max\{2 \cdot S, S_1\}$ hoch.

- **Slow Start** lässt die anfänglich sehr geringe Emissionsrate multiplikativ wachsen (**multiplicative increase**) bis entweder der Schwellenwert S_1 überschritten wird oder ein Fehler auftritt.
- Wird S_1 überschritten, dann wechsle zu **Congestion Avoidance**:
  - Erhöht die Fenstergröße S um Eins (**additive increase**), wann immer alle Pakete eines Fensters bestätigt wurden.

Fast Retransmit und Fast Recovery

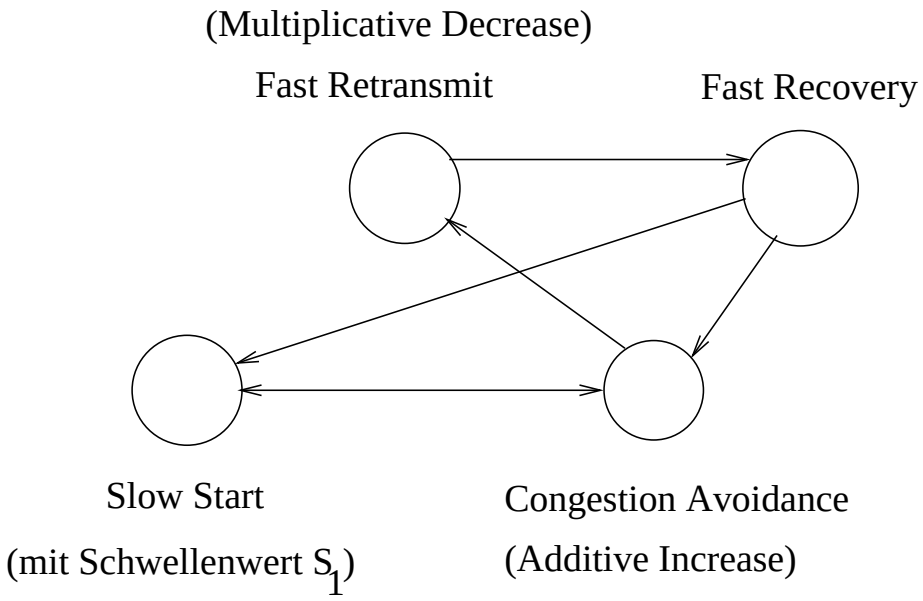
Fast Retransmit wird aufgerufen, wenn ein Ordnungsfehler für ein Paket auftritt.

- Das (vermutlich) verlorene Paket wird neu versandt ohne den Retransmission Timeout abzuwarten.
- Die gegenwärtige Emissionsrate S wird halbiert (**multiplicative decrease**) und **Fast Recovery** wird aufgerufen.

Fast Recovery

- Mit Hilfe der selektiven Bestätigungen wird ein *Scoreboard* berechnet, das alle vermutlich verlorenen Pakete des Fensters aufführt.
- Der Sender schätzt die Anzahl der (regulär) ausstehenden Pakete und hält die Schätzung in der Variable „pipe“ fest.
- Der Sender sendet Pakete des Scoreboards bzw. neue Pakete nur dann, wenn $\text{pipe} < S$ für die aktuelle Fenstergröße S gilt.
  - pipe wird für jedes versandte Paket um Eins erhöht und für jede Bestätigung um Eins erniedrigt: Die Belastung des Netzes ist somit durch S beschränkt.
- Tritt ein RTO-Fehler auf, wird **Slow Start** aufgerufen, wobei die aktuelle Fenstergröße als neuer Schwellenwert S_1 gewählt wird.
- Werden hingegen alle Pakete des Scoreboards bestätigt, erfolgt ein Aufruf von **Congestion Avoidance**.

Die vier Komponenten



Die Kombination von Congestion Avoidance und Fast Retransmit folgt somit dem konservativen Prinzip des
additiven Anwachsen, multiplikativen Fallens (AIMD).

Active Queue Management (AQM)

AQM-Algorithmen entfernen sogar vorsätzlich Pakete als Teil einer Verkehrskontrolle.

- Warum entfernt man Pakete, wenn dies nicht erzwungen ist?
- Der Sender stellt fest, dass Pakete nicht ankommen und vermindert daraufhin (hoffentlich) die Sendegeschwindigkeit.

- Der AQM-Algorithmus **Drop-Tail**:
  - Ankommende Pakete werden abgewiesen, wenn die maximale Queuegröße überschritten wird.
  - Paketstaus werden nur langsam aufgelöst, da Sender zu spät von überlaufenden Queues erfahren.
- Der AQM-Algorithmus **RED** (Random Early Detection):
  - RED arbeitet mit dem Schwellenwert *MIN* und *MAX*.
  - Bei gegenwärtiger Queuegröße *M* werden Pakete mit Wahrscheinlichkeit p_M entfernt: Fine-Tuning von p_M ist erforderlich.

Active Queue Management: Blue

Pakete werden zufällig durch RED entfernt:

+ Die Aufteilung der Bandbreite ist fair.
- Eine Vielzahl von Verbindungen wird zur Neuübertragung oder zum Abbremsen veranlasst, was den Stau verschärfen kann.
- Fine-Tuning von p_M ist erforderlich.

- Der AQM-Algorithmus **Blue**:
  - Die Löschwahrscheinlichkeit wird an die Queuegröße gekoppelt.
  - Blue erhöht die Löschwahrscheinlichkeit bei Überlast auf $p \cdot (1 + \varepsilon)$ und reduziert bei Unterlast auf $p \cdot (1 - \varepsilon)$.
- Der AQM-Algorithmus **Stochastic Fair Blue** (SFB):
  - Die Rate von Paketflüssen wird durch Hashing oder die speicher-effizienteren Bloom-Filter approximiert. Pakete eines Flusses werden entsprechend ihrer Rate zufällig gelöscht.
  - Reagiert ein aggressiver Fluß nicht durch Ratenreduktion, wird er bestraft.

AIMD

Kampf um eine Ressource

n Verbindungen kämpfen um die Ressource „Bandbreite“.

- Verbindung i habe zum Zeitpunkt t den Anteil $x_i(t) \geqslant 0$ der Ressource angefordert.
- Das System antwortet mit dem Befehl **„fahr hoch“** oder **„fahr runter“**, der von allen Verbindungen zu befolgen ist.
- Bestimme einen Anforderungsmechanismus

$$x_i(t+1) = \begin{cases} a_H + b_H \cdot x_i(t) & y_t = \text{fahr hoch}, \\ a_R + b_R \cdot x_i(t) & y_t = \text{fahr runter} \end{cases}$$

so dass der Anstieg $F_{t+1} - F_t$ der Fairness zum Teitpunkt $t+1$ maximiert wird.

$$F_t = \frac{(\sum_{i=1}^{n} x_i(t))^2}{n \cdot \sum_{i=1}^{n} x_i(t)^2}$$

ist die Fairness zum Zeitpunkt t.

Fahr runter!

Verbindung i muss ihre Anforderung für „$y_t = \text{fahr runter}$” drosseln.

- Wir müssen $x_i(t+1) = a_R + b_R \cdot x_i(t) < x_i(t)$ fordern, falls $x_i(t) > 0$ gilt.
- Stelle $x_i(t+1) \geqslant 0$ auch für kleine Werte von $x_i(t)$ sicher:
  a_R ist nicht-negativ.
- $x_i(t+1) < x_i(t)$ muss für kleine Werte von $x_i(t)$ garantiert werden:
  $a_R = 0$ folgt.

Es ist $a_R = 0$ und $0 \leqslant b_R < 1$.

Fahr hoch!

Wir müssen $x_i(t+1) = a_H + b_H \cdot x_i(t) > x_i(t)$ garantieren und erhalten die Bedingungen

$$a_H \geqslant 0 \quad \text{und} \quad b_H \geqslant 1.$$

Wenn F_t die Fairness zum Zeitpunkt t ist:

$$\begin{aligned} F_{t+1} &= \frac{(\sum_{i=1}^n x_i(t+1))^2}{n \cdot \sum_{i=1}^n x_i(t+1)^2} = \frac{(\sum_{i=1}^n a_H + b_H \cdot x_i(t))^2}{n \cdot \sum_{i=1}^n (a_H + b_H \cdot x_i(t))^2} \\ &= \frac{(\sum_{i=1}^n a_H/b_H + x_i(t))^2}{n \cdot \sum_{i=1}^n (a_H/b_H + x_i(t))^2} = G(c) \end{aligned}$$

mit $c = a_H/b_H$ und $G(c) = (\sum_{i=1}^n c + x_i(t))^2/(n \cdot \sum_{i=1}^n (c + x_i(t))^2$.

Analyse (1/2)

- $F_{t+1} = G(c) = (\sum_{i=1}^{n} c + x_i(t))^2 / (n \cdot \sum_{i=1}^{n} (c + x_i(t))^2$.
- G ist monoton wachsend: Zeige $G'(c) \geqslant 0$ für $c \geqslant 0$.

- $(\frac{f}{g})' \geqslant 0$ ist äquivalent zu $f' \cdot g \geqslant f \cdot g'$.
- Zeige $2n \cdot (\sum_{i=1}^{n} c + x_i(t)) \cdot (n \cdot \sum_{i=1}^{n} (c + x_i(t))^2) \geqslant (\sum_{i=1}^{n} c + x_i(t))^2 \cdot (2n \cdot \sum_{i=1}^{n} (c + x_i(t)))$.
- Die Ungleichung ist äquivalent zu

$$n \cdot \sum_{i=1}^{n} (c + x_i(t))^2 \geqslant (\sum_{i=1}^{n} c + x_i(t))^2.$$

Die Ungleichung ist stets erfüllt, denn

$$n \cdot \sum_{i=1}^{n} a_i^2 \geqslant (\sum_{i=1}^{n} a_i)^2.$$

Analyse (2/2)

- G ist monoton wachsend für $c \geqslant 0$.
- Wähle c größtmöglich,
  um einen maximalen Fairnessanstieg zu erreichen.

- Wenn der Befehl „Fahr runter“ ist,
  dann bleibt die Fairness unverändert, denn $a_R = 0$.
- Und wenn der Befehl „Fahr hoch“ ist?
  - Für $a_H = 0$ bleibt die Fairness unverändert, für $a_H > 0$ kann die Fairness nur zunehmen.
  - Wir wissen, dass $b_H \geqslant 1$ gilt und dass $c = a_H / b_H$ größtmöglich zu wählen ist.

Um die Fairness zu maximieren, wähle AIMD

- $a_H > 0$, $b_H = 1$ (additive increase) und
- $a_R = 0, b_R < 1$ (multiplicativ decrease).

Und die Konsequenzen für TCP

- Ein additives Anwachsen führt zum stärksten Fairnessanstieg:

  Ist das überraschend? Nicht wirklich,
  jede beteiligte Verbindung erhält denselben Zusatzgewinn.

- Wir haben aber keine wirklich überzeugende Rechtfertigung des **additive increase - multiplicative decrease** Rezepts erhalten:

  Wir haben uns auf die kleine Klasse der linearen Anforderungsschemata eingeschränkt.

Ein dynamisches Modell

- Verbindungen kämpfen wiederum um die Ressource „Bandbreite“.
- Alte Verbindungen terminieren möglicherweise und neue Verbindungen kommen hinzu.

Wie schnell konvergiert AIMD gegen ein sich jetzt dynamisch änderndes Fairness-Optimum,

wenn die Bandbreite im Hochfahren um 1 erhöht und im Runterfahren um den Faktor $\beta < 1$ erniedrigt wird?

Die Bandbreite einer neuen Verbindung

- Wir fixieren eine Verbindung V und nehmen an, dass das System eine Gesamtbandbreite B zur Verfügung stellt.
- V wird zum Zeitpunkt T mit Bandbreite 0 aufgenommen und wird dann in einer Folge von Additionsschritten Bandbreite gewinnen.
  - $T_0 < T_1 < \cdots T_k < \cdots$ sei die Folge der Zeitpunkte, zu dem das System den Befehl des Runterfahrens gibt.
  - Es gelte $T \leqslant T_0$.

- Wenn Verbindung V bis zum Zeitpunkt T_k besteht, dann beträgt ihre Bandbreite vor dem Zeitpunkt T_k mindestens

$$(1-\beta)^k \cdot \frac{B}{n_k},$$

  - n_k ist die Anzahl der Verbindungen zum Zeitpunkt T_k.
- Zu einem Zeitpunkt $t \in [T_{k-1}, T_k[$ besitzt jede andere Verbindung höchstens die Bandbreite von V plus die Bandbreite $\beta^k \cdot B$.

Gerechtfertigte Bandbreite

- Die von den einzelnen Verbindungen zum Zeitpunkt T gehaltenen Bandbreiten sind **ungerechtfertigt**.
  - Wir nehmen die Sichtweise der Verbindung V ein, die zum Zeitpunkt T mit Bandbreite 0 beginnen muss.
- Die nach dem Zeitpunkt T vergebenen Bandbreiten sind **gerechtfertigt**.
  - Die Verbindung V muss die ihrerseits gewonnene Bandbreite auch den anderen Verbindungen zugestehen.
- Wieviel Bandbreite ist zum Zeitpunkt T_k gerechtfertigt?
  - Nur die Bandbreite zum Zeitpunkt T ist ungerechtfertigt und danach kommt nie mehr ungerechtfertigte Bandbreite hinzu.
  - Die ungerechtfertigte Bandbreite zu jedem Zeitpunkt T_i wird um den Faktor β erniedrigt.

Vor dem Zeitpunkt T_k
ist höchstens die Bandbreite $\beta^k \cdot B$ ungerechtfertigt.

Berechtigte Bandbreite von V

- Keine andere Verbindung besitzt mehr berechtigte Bandbreite als V: Spätere Verbindungen besitzen weniger und zwischenzeitlich terminierende Verbindungen geben zusätzliche Bandbreite frei.

V hat vor dem Zeitpunkt T_k
mindestens die Bandbreite $(1 - \beta^k) \cdot B/n_k$ errungen.

- Wieviel Bandbreite besitzen andere Verbindungen höchstens?
  - Eine andere Verbindung W wird dann die meiste Bandbreite an sich reißen, wenn sie zum Zeitpunkt T die gesamte (und damit ungerechtfertigte) Bandbreite B besitzt.
  - Zu einem Zeitpunkt $t \in [T_{k-1}, T_k[$ ist aber die ungerechtfertigte Bandbreite auf $\beta^k \cdot B$ geschmolzen.
  - Die Verbindung W besitzt höchstens die (gerechtfertigte) Bandbreite von V plus die ungerechtfertigte Bandbreite $\beta^k \cdot B$.

Das Steady State Modell

Das Steady State Modell für AIMD

1. Ein Sender fährt seine Sende-Rate um jeweils ein Paket solange hoch, bis ein Schwellenwert S erreicht ist.
2. Danach erfolgt ein Paketverlust und der Sender muss seine Rate von S auf $\frac{S}{2}$ halbieren.

Berechne den Durchsatz D in Abhängigkeit von der Verlustrate p.

- Beginnend mit Senderate $\frac{S}{2}$ erhalten wir in den ersten $S/2$ Schritten den Gesamtdurchsatz $D^* = \sum_{i=S/2}^{S} i \approx \frac{3}{8} \cdot S^2$.
- D^* fällt auch in jedem folgendem Zeitintervall der Länge $\frac{S}{2}$ an.

Also erhalten wir den durchschnittlichen Durchsatz
$D \approx \frac{3}{8} \cdot S^2/(S/2) = \frac{3}{4} \cdot S$.

Durchsatz und Verlustrate im Steady State Modell

In jedem Zeitintervall der Länge $S/2$ gibt es genau einen Paketverlust.

- $p \approx \frac{8}{3 \cdot S^2}$ ist die durchschnittliche Verlustrate.
- Wir haben die Beziehung

$$D = \Theta(\frac{1}{\sqrt{p}})$$

  erhalten.
- **Übungsaufgabe**:

  Zeige, dass die Beziehung $D = \Theta(\frac{1}{p})$ für AIAD, MIAD und MIMD gilt: AIMD besitzt den höchsten Durchsatz als Funktion der Verlustrate.

AIMD: Kritikpunkte

1. TCP ändert Geschwindigkeiten nur aufgrund von Paket-Verlusten.
   - Paket-Verlust ist ein binäres Signal, das nicht den Grad der Verstopfung angibt.
     - ★ Eine der Ursache angepasste Reaktion ist nicht möglich.
     - ★ Vorbeugende Gefahrmeldungen durch Router scheinen vernünftiger als ein Warten auf den Unfall.
   - Verfahren des Active Queue Management geben aber implizit das Ausmaß der Verstopfung wieder.
2. Die Reaktionszeit hängt stark von der Round-Trip Time ab, die kontroll-theoretisch als Feedback-Verzögerung wirkt.
   - TCP benachteiligt Paketflüsse mit grosser Round-Trip Time.
3. Slow Start benötigt $\Theta(\log_2 N)$ Verdopplungen bis die volle Bandbreite N erreicht ist.
   - Kurzlebige Flüsse verlassen Slow Start während ihrer Lebenszeit nicht: Schlechte Ausnutzung der möglichen Bandbreite.

Alternative Ansätze

HighSpeed TCP

- verhält sich wie das konventionelle TCP, wenn das Sliding Window nicht zu groß ist.
- Auch für große Sliding Windows wird das AIMD Verfahren beibehalten, allerdings mit aggressiveren Inkrementen und langsamerem Abbremsen.
- Slow Start wird durch Limited Slow Start ersetzt, wenn TCP mit einem großen Sliding Window beginnt:
  - Statt die Senderate exponentiell zu steigern, wird die Fenstergröße nach jedem Acknowledgement um eine Konstante vergrößert.
  - Selbst wenn ein Acknowledgement ausbleibt, wird die Fenstergröße allerdings um eine kleinere Konstante vergrößert.

QuickStart

Eine Verbindung zwischen Hosts *A* und *B* soll eröffnet werden.

- *A* sendet ein Paket mit einer gewünschten Emissionsrate.
  - Router auf dem Weg von *A* nach *B* können die gewünschte Emissionsrate billigen, modifizieren oder ablehnen.
  - Host *B* sendet dann das Ergebnis der “Router-Umfrage” an *A* zurück.
- Sollte die Anfrage abgelehnt werden, wählt *A* eine Default-Emissionsrate.

Natürlich ist das Ziel, bereits mit einer großen Emissionsrate, beziehungsweise mit einem großen Sliding Window zu beginnen.

Explicit Control Protocol (XCP)

- XPC verschickt *explizite* Stauinformationen:
  - Die Router informieren Sender und Empfänger durch Acknowledgements über den Verstopfungsgrad und erlauben damit eine der Ursache angepasste Reaktion.
- Router behandeln Effizienz (möglichst große Ausnutzung der Bandbreite) und Fairness (möglichst gleichmäßige Aufteilung der Bandbreite) getrennt.
  - Das Ziel: Eine schnellere, aber faire Bandbreitennutzung.

1. QuickStart ist ein Teil von XPC:
   - Der Sender *A* gibt die Größe des Sliding Windows, die Round-Trip Time und die gewünschte Emissionsrate im Paket-Header bekannt.
   - Jeder Router auf dem Weg kann die Emissionsrate modifizieren.
   - Der Empfänger *B* verschickt dann die Empfehlung als Teil des Acknowledgements zurück an *A*.

XCP

2. Jeder Router R bestimmt für jeden Link die durchschnittliche Round-Trip Time T der über den Link führenden Pakete.
   - R wartet ein Zeitintervall der nächsten T Schritte ab.
   - Danach wird ein neues Zeitfenster mit aktualisiertem T aufgemacht und das Vorgehen wiederholt sich.
3. Der Effizienz- und Fairness-Controller geben in jedem Zeitintervall eine Geschwindigkeitsempfehlung ab.
   - Der Effizienz-Controller bestimmt am Ende des alten Zeitfensters
     - ★ die freie Bandbreite B und die minimale Queuegröße Q
     - ★ und setzt $\phi = \alpha \cdot B - \beta \cdot Q$ als zu vergebende / drosselnde Bandbreite für Koeffizienten $\alpha, \beta \in [0, 1]$.
     - ★ Q geht negativ ein, um die Queue zu entleeren.
   - Der Fairness-Controller wendet AIMD an, um ϕ auf die Flüsse aufzuteilen:
     - ★ Für $\phi \geqslant 0$ erhält jeder Fluss dasselbe additive Inkrement.
     - ★ Für $\phi < 0$ sinkt die Bandbreite jedes Flusses um denselben Faktor.