Chapter 13

Advanced computer networking

Internet Protocols

Thomas Fuhrmann

Network Architectures
Computer Science Department
Technical University Munich

Dienstgüte – Quality of Service (QoS)

- Das Internet wurde entworfen, um einen Ende-zu-Ende Dienst auf Best-Effort-Basis bereit zu stellen.
  - Datagramme kommen am Ziel an oder werden ggf. verworfen
  - TCP stellt sicher, dass ein Datenstrom korrekt am Ziel ankommt, macht aber keine Aussage ob die Daten kontinuierlich ankommen.
- Für „Multimedia“ Kommunikation (besser zeitkontinuierliche Medien) reicht dieser Dienst nicht aus.

- ISO Definition:
  - „Dienstgüte (engl. Quality of Service) bezeichnet das definierte, kontrollierbare Verhalten eines Systems bezüglich quantitativ messbarer Parameter.”
- Dienstgüte betrifft die Präsentation und Übertragung von kontinuierlichen und diskreten Informationen.
  - Insbesondere Echtzeitanforderungen (engl. real time)
  - Aber auch Kosten, Sicherheit (Geheimhaltung & Authetifizierung, engl. security), Ausfallsicherheit (Zuverlässigkeit, engl. Safety), etc.

Echtzeitbetrieb

- Definition Echtzeitbetrieb (nach DIN 44300):
  - „Echtzeitbetrieb ist ein Betrieb eines Rechensystems, bei dem Programme zur Bearbeitung anfallender Daten ständig derart betriebsbereit sind, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind."
- Korrektheit der erbrachten Ergebnisse erfordert *fehlerfreie Berechnung* und Einhalten des *Zeitpunkts der Bereitstellung* des Ergebnisses.
- Garantierte Antwortzeiten:
  - Hohe Geschwindigkeit ist nicht zwingend Hauptmerkmal.
  - Zu schnell kann manchmal auch nicht akzeptabel sein um Synchronisation zwischen Multimedia-Strömen zu gewährleisten.

Allgemeine Echtzeitanforderungen

- Allgemeine Anforderungen an Echtzeitsysteme:
  - Multitasking-Fähigkeit
  - Kurze Unterbrechungs-verzögerung
  - Schnelle Kontextwechsel
  - Kontrolle der Speicherverwaltung
  - Geeignetes Scheduling
  - Zeitgeberdienste (Timer) einer feinen Granularität
  - Geeignete Interprozesskommunikations- und Synchronisationsmechanismen
- Erfordert Unterstützung durch entsprechendes Echtzeit-betriebssystem (Real Time Operating System = RTOS).

- Zeitschranke:
  - Definiert den letzten Zeitpunkt, zu dem das Ergebnis einer Berechnung noch korrekt ist.
  - Grenze zwischen korrektem und fehlerhaften Verhalten.
- **Weiche Zeitschranken** (stochastische Anforderungen):
  - Orientierungspunkte, die eine gewisse Abweichung zulassen
  - Beispiel: “In 90% aller Fälle müssen Ergebnisse innerhalb eines bestimmten Zeitraums vorliegen.”
- **Harte Zeitschranken**:
  - Zeitschranken, die unbedingt eingehalten werden müssen.
  - Beispiel: Auslösen der Bremsen bei elektronischen Fahrhilfen im Auto.

Anforderungen von Multimediasystemen

Echtzeitanforderungen an bzw. von Multimediasystemen:

- Audio-/Videokommunikation durch Zeitschranken begrenzt, d.h. Daten müssen zum Abspielzeitpunkt vorliegen. (Allerdings sind die Anforderungen aufgrund von Fehlertoleranz meist nicht so strikt wie bei klassischen Echtzeitsystemen.)
- Möglichst geringe Verzögerungsschwankungen (engl. Jitter), d.h. alle Datagramme sollten möglichst gleich lange im Netz unterwegs sein.
- Bei interaktiven Anwendungen auch möglichst geringe Verzögerung.
- Synchronisation

Weitere Anforderungen

- Hoher Datendurchsatz, insbesondere bei der Videokommunikation
- Skalierbarkeit durch z.B. unterschiedliche Qualitäten
- Fairness bezüglich der Aufteilung von Ressourcen für zeitkritische und nicht-zeitkritische Anwendungen

Dienstklassen

Bestmögliche Klasse (engl. Best-Effort):

- Typische Dienstklasse im Internet – in der Praxis bis heute!
- Keine Ressourcenreservierung, sondern mit den verfügbaren Ressourcen so gut (und schnell) umgehen, wie möglich.
- Konflikte zwischen Anwendungen und Anwendern sind zu erwarten

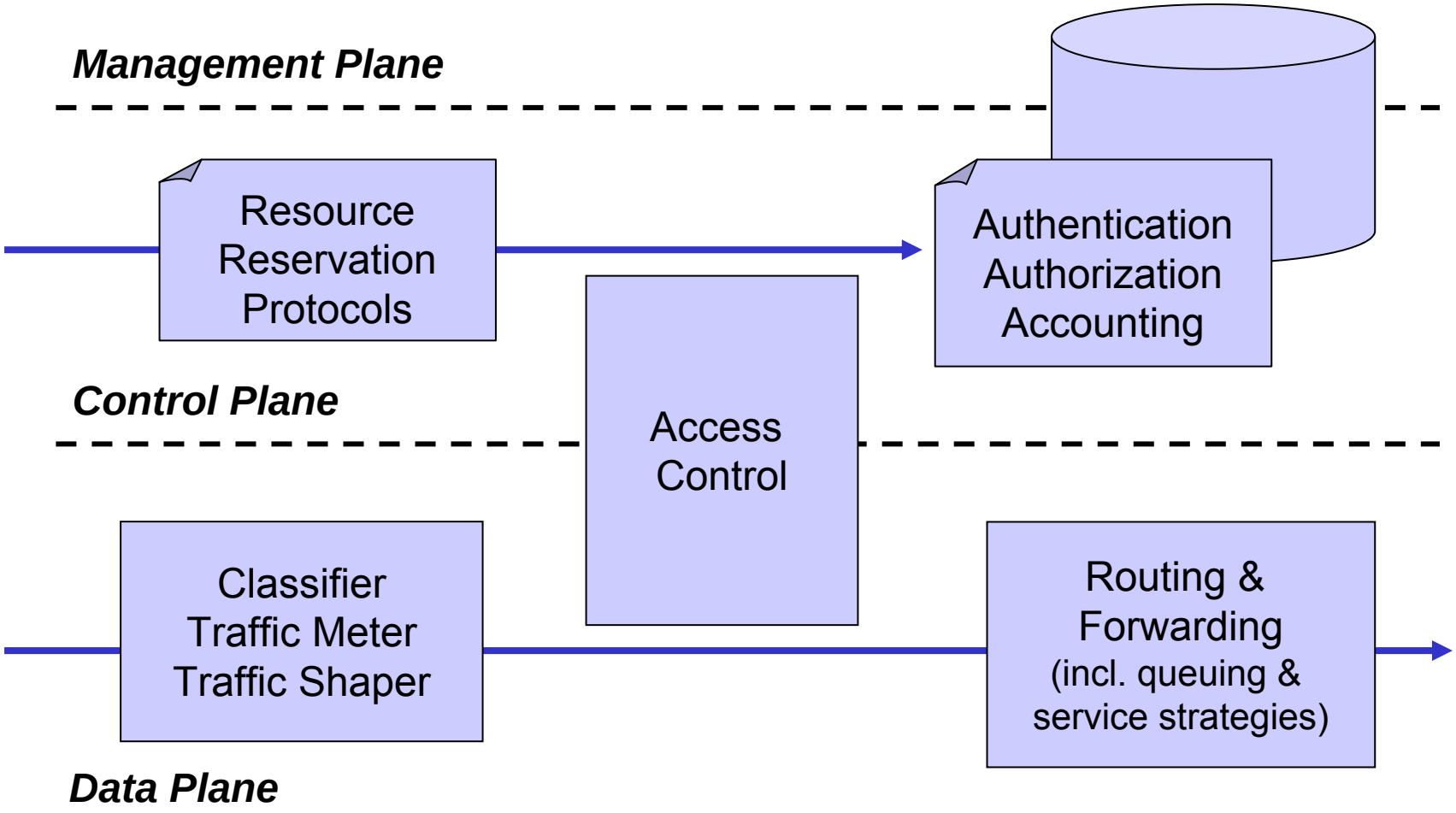
Statistische Klasse:

- Dienstgüteparameter werden nur statistisch eingehalten z.B.: „80% der Pakete haben eine Verzögerung < 100ms“
- Ressourcen werden bis zu einem gewissen Grad überbelegt
- Konflikte sind immer noch möglich

Deterministische Klasse:

- Dienstgüteparameter werden immer eingehalten (harte Garantie)
- Ressourcen stehen einem Nutzer exklusiv zur Verfügung
- Garantiert keine Konflikte wenn die Ressource gewährt wurde
- Somit aber auch „Besetztfall“ bei der Ressourcenanfrage wenn keine Ressourcen mehr übrig sind

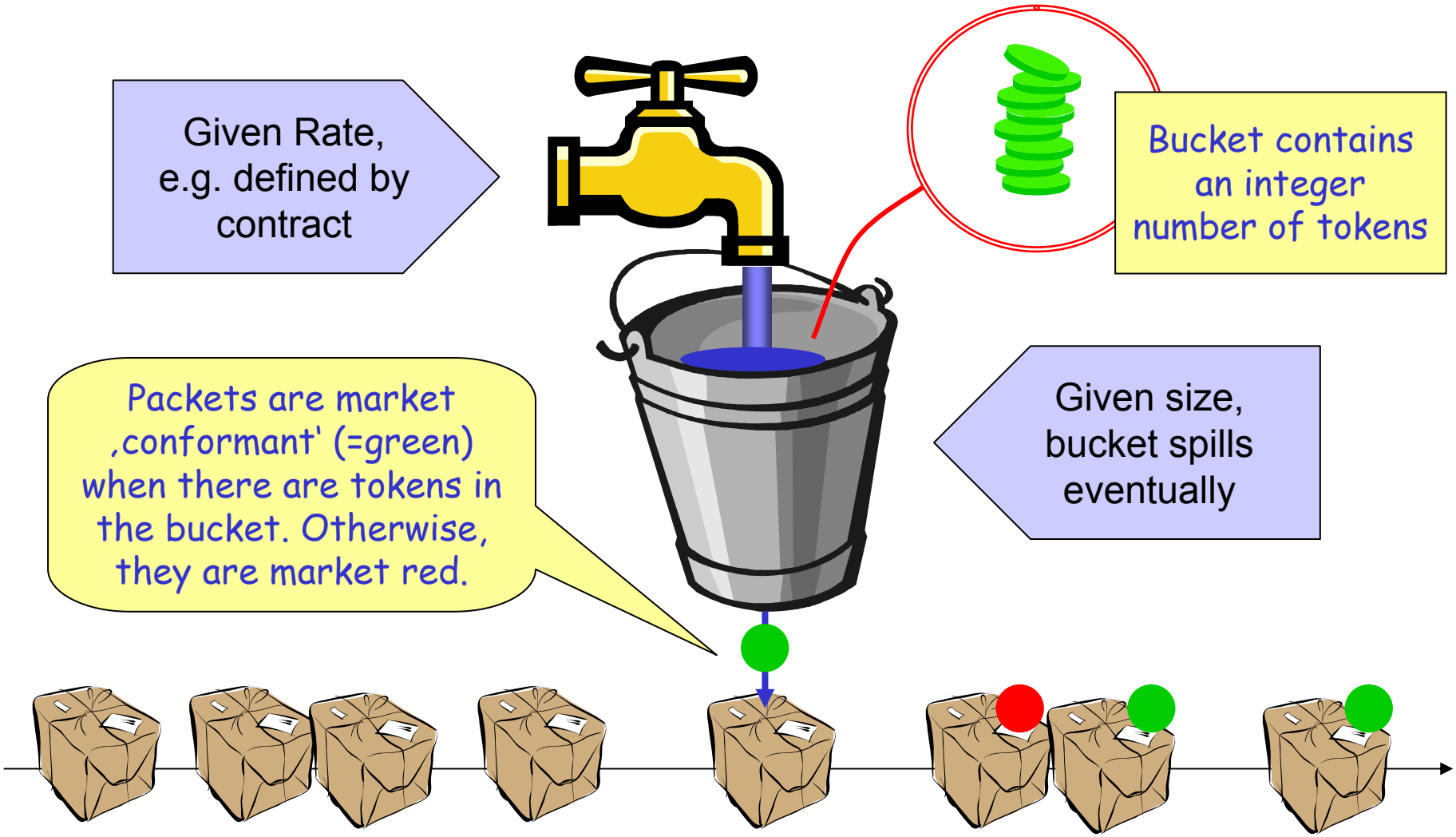
The QoS Landscape



Komponenten im Datenpfad

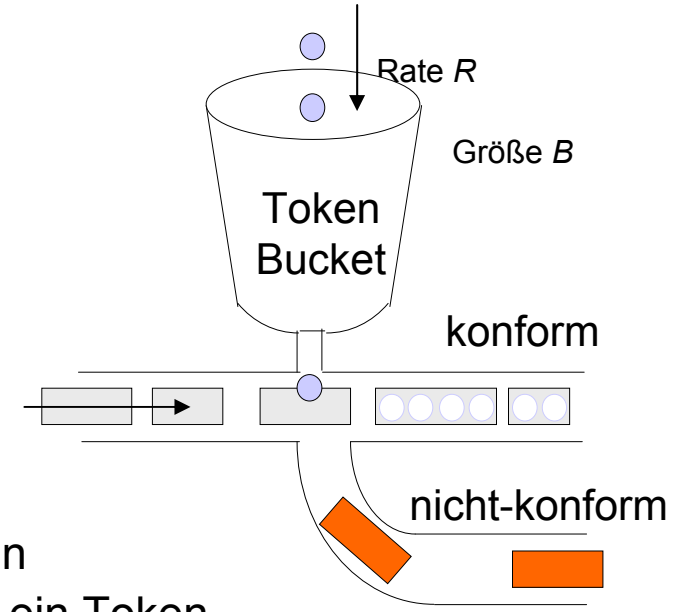
- Verkehrsmeter (engl. Traffic Meter)
  - Prüfen der Datenpakete auf Konformität zum Verkehrsvertrag (Nutzungskontrolle)
- Verkehrsformer (engl. Traffic Shaper)
  - Ändern der Charakteristik eines Datenstroms/Aggregats
  - Wiederherstellen von Konformität
- Klassifikationselemente (engl. Classifier)
  - Einteilung von einzelnen Datenpaketen in Datenströme
  - Datenströme: Einheiten auf denen Dienstgütemechanismen arbeiten
- Warteschlangen
  - Unterschiedliche Behandlung von Datenströmen
- Bedienstrategien für Warteschlangen
  - Zuteilung der Ressourcen (z. B. Bandbreite, Rechenzeit, Speicher) durch Scheduler

Traffic Meter & Classifier – Token Bucket

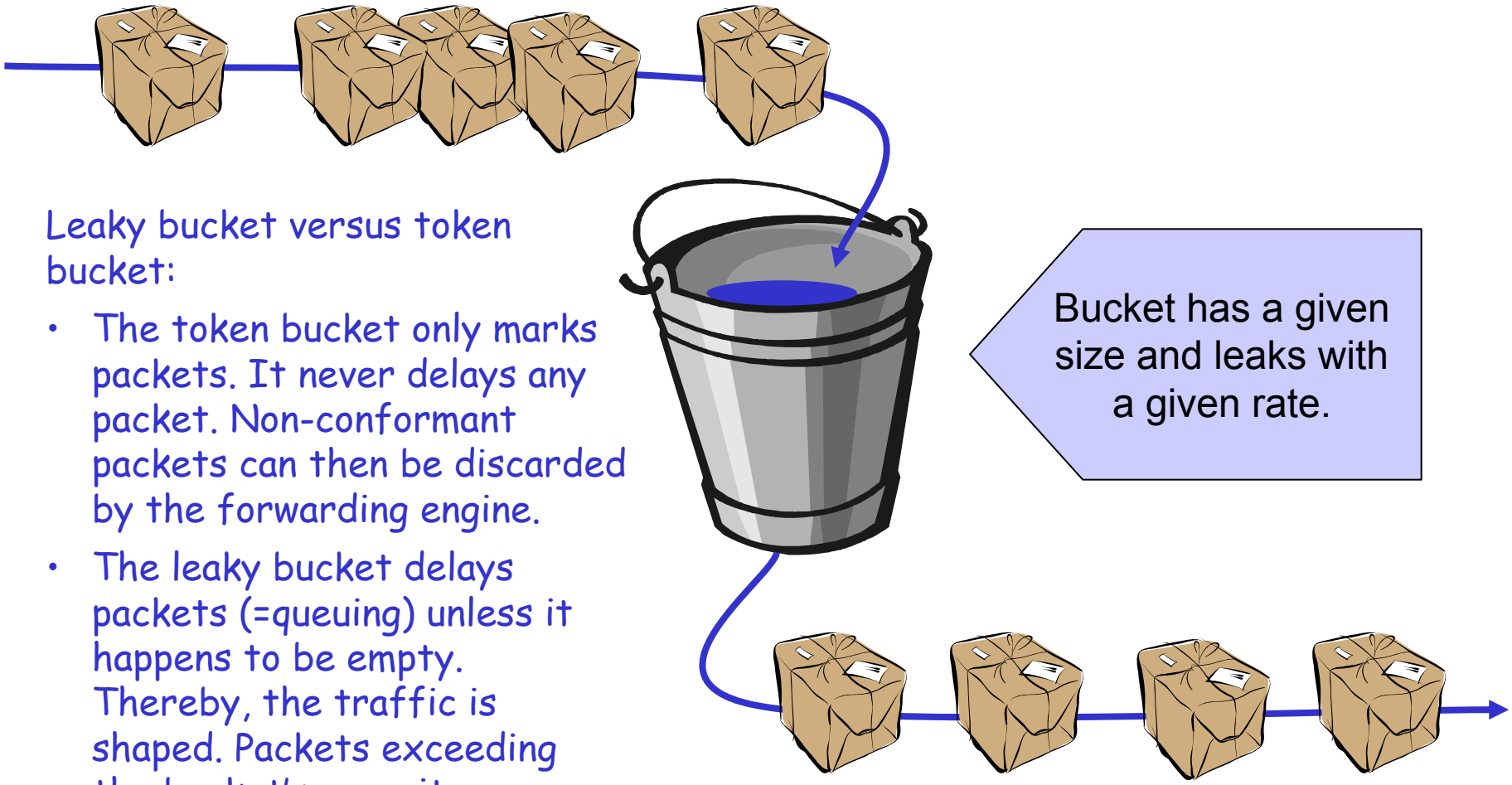


Beispiel für Verkehrsmeter: Token Bucket

- Eimer mit Größe B Token (***Senderechte***)
- Token „tröpfeln“ mit der Rate R in den Eimer
- maximal B Token im Eimer (Token laufen über)
- überwacht die Einhaltung einer Rate R (bit/s) mit einer Toleranz (Burst) B (byte)
- Pakete werden als ***konform*** markiert, wenn bei Ankunft eines Pakets noch genügend Token im Eimer vorhanden sind (oft: 1 Byte = 1 Token)
  - Strenge Variante: für Paket mit Länge L müssen mindestens L Token vorhanden sein
  - Lockere Variante: Bucket muss mindestens ein Token enthalten, fehlende Tokens werden geborgt, d.h. Tokenstand wird negativ
- Andernfalls wird ein ankommendes Paket als ***nicht-konform*** markiert
- Der Token-Bucket misst nur den Verkehr, es erfolgt keine Formung – Bursts bleiben erhalten (sofern genügend Token vorhanden sind)
- Token Bucket wird oftmals konzeptionell mit unendlicher Rate geleert
- Effektive Burst-Größe B_E kann größer als B sein, wenn Token Bucket mit endlicher Rate (max. Rate des Netzwerkadapters) geleert wird (da während des Leerens neue Token generiert werden).



Traffic Shaper – Leaky Bucket



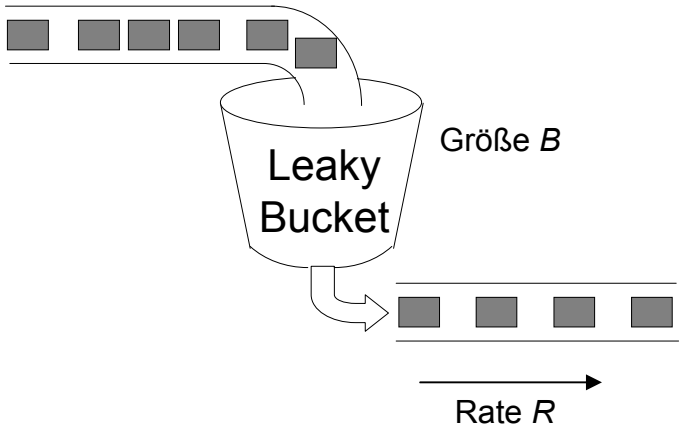
Leaky bucket versus token bucket:

- The token bucket only marks packets. It never delays any packet. Non-conformant packets can then be discarded by the forwarding engine.
- The leaky bucket delays packets (=queuing) unless it happens to be empty. Thereby, the traffic is shaped. Packets exceeding the bucket's capacity are marked or discarded.

Beispiel für Verkehrsformer: Leaky Bucket

Leaky-Bucket:

- Eimer mit max. Größe B Byte
- Pakete werden in den Eimer gepackt und „tröpfeln“ mit der Rate R aus dem Eimer heraus
- Überwacht die Einhaltung einer Rate R (bit/s) mit einer Toleranz (Burst) B (byte)
- ***Glättet*** den Ausgangsstrom auf die Rate R
- Ein Paket der Größe L wird gesendet, wenn L Byte aus dem Eimer getropft sind

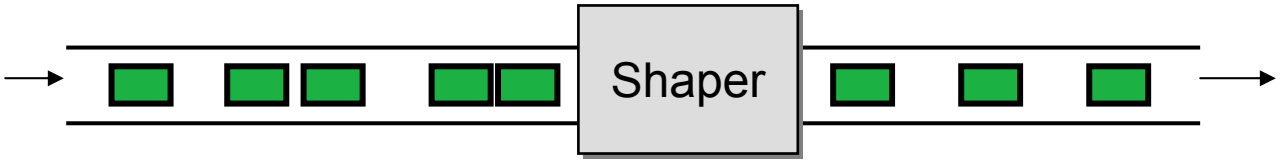


- Hat ein neu eintreffendes Paket keinen Platz mehr im Eimer wird es verworfen oder degradiert
- Pakete im Eimer werden so lange verzögert, bis das Senden konform zur Rate ist (d.h. ggf. zusätzliche Verzögerung)

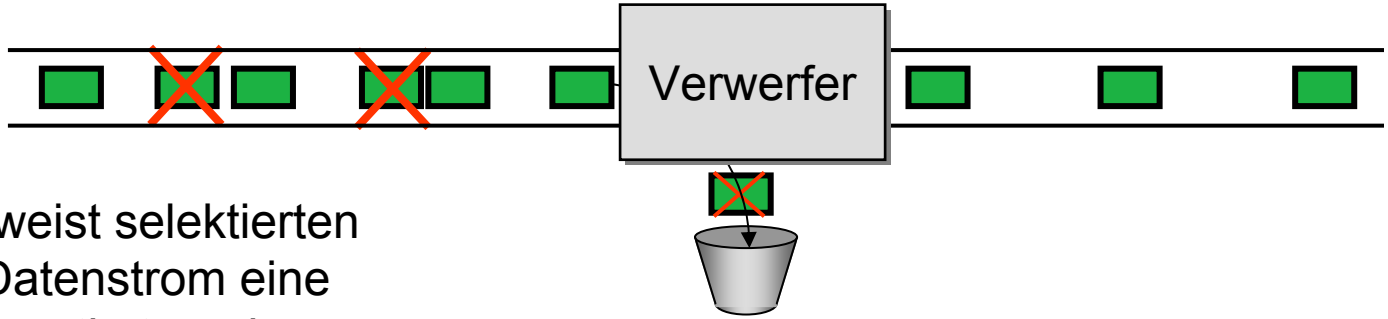
Verkehrsformer

Verkehrsformer beeinflussen aktiv die Charakteristik des Verkehrs:

- Wiederherstellung von Konformität bzw. eines bestimmten Verkehrsmusters
- Verkehrsglätter glättet alle Pakete eines Datenstroms (Shaper):

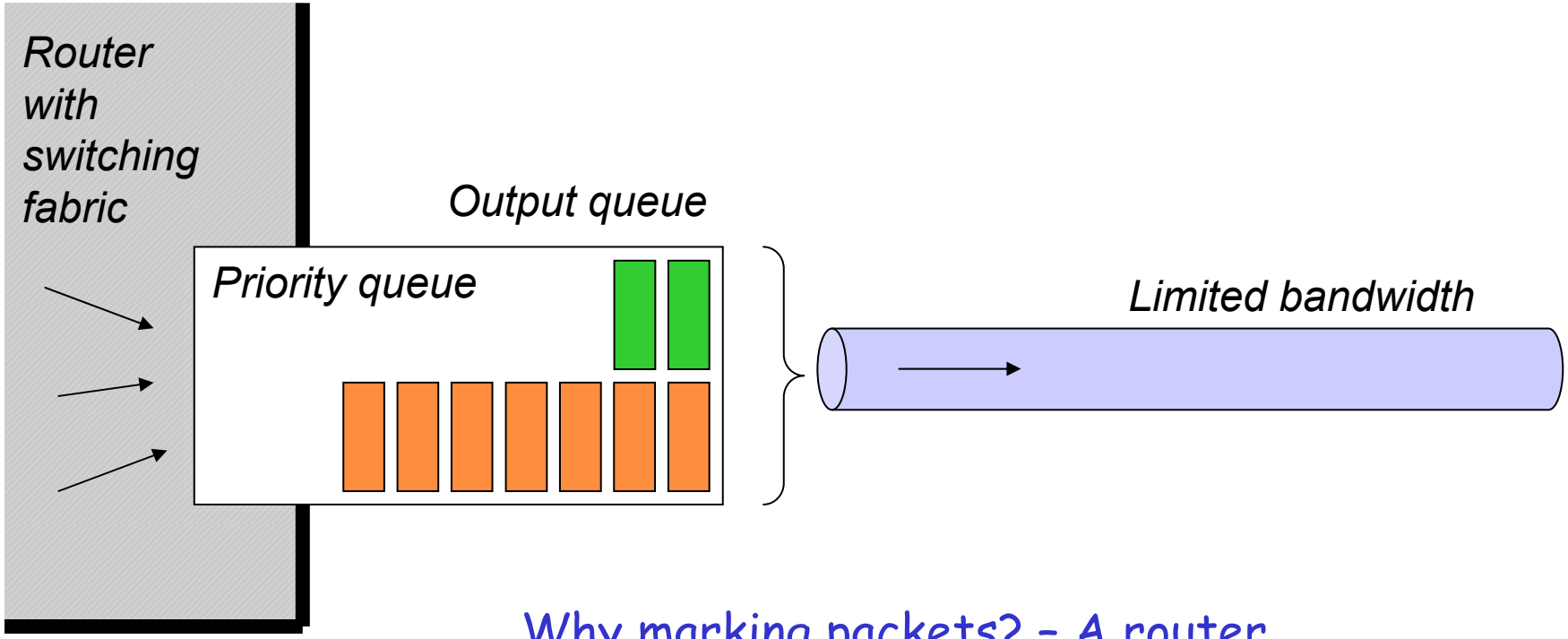


- Verwerfer selektiert zu verwerfende Pakete:



- Degradierer weist selektierten Paketen im Datenstrom eine niedrigere Dienstkategorie zu

Why marking packets?



Why marking packets? – A router output queue can then, for example, forward green packets with high priority, and send red packets only when there are no green packets in the queue.

Fairness for Whom?

- We are all individuals
  - Every connection has a right for its own resources.
  - Requires state to distinguish between the traffic flows

- We are in one boat
  - Traffic is aggregated
  - Different aggregates are treated differently
  - More “keep it simple & stupid” than treat everyone with their own rights

Warteschlangen und Bedienstrategien

Warteschlangen speichern Pakete:

- FIFO - First In First Out
- EDF - Earliest Deadline First
- Prioritäten, d.h. verschiedene logische Warteschlangen

Bedienstrategien bestimmen die nächste zu bedienende Warteschlange bzw. das nächste zu bedienende Paket:

- Legt die Reihenfolge der exklusiven Ressourcennutzung (Bandbreite, Speicher) abhängig vom Scheduling-Algorithmus fest
- Scheduler überall erforderlich, wo gemultiplext wird

Typische Verfahren:

- Round Robin (Reihum): alle nacheinander - keiner wird bevorzugt
- Simple Priority Queueing, d.h. immer erst hochpriore Warteschlange bedienen (=permanente Bevorzugung)
- Weighted Fair Queueing: Round Robin mit Gewichtung, d.h. Warteschlange mit hohem Gewicht wird öfter bedient

Anforderungen an Scheduling-Algorithmen

Scheduling-Algorithmen sollen einfach sein.

- Scheduling geschieht im Datenpfad und muss daher schnell ausführbar sein.
- Dies gilt für Best-Effort-Dienste wie auch für garantierte Dienste.
- Insbesondere soll der Aufwand möglichst O(1) und nicht O(N) sein, wenn N für die Anzahl der Datenströme steht
- Speicheranforderungen zur Zustandshaltung sollten möglichst gering sein, d.h. Zugriffe auf den Zustand müssen effizient möglich sein.

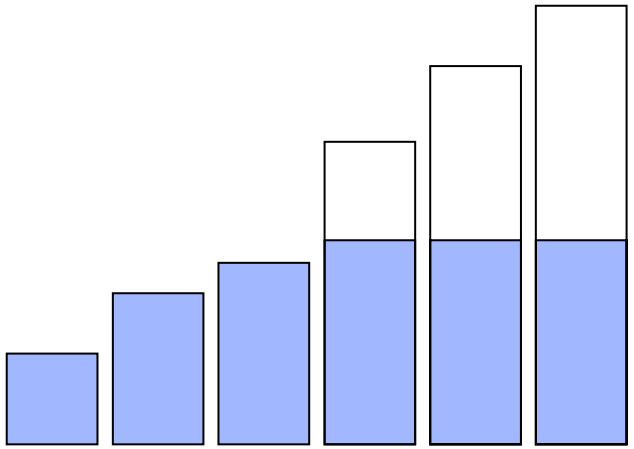
Scheduling-Algorithmen sollen fair sein.

- Insbesondere bei Best-Effort-Diensten, aber auch bei garantierten Diensten.
- Was ist Fairness? – Beispiel: Wenn „Max-min Fair Share“ (siehe unten) eingehalten wird, dann liegt ***Fairness*** vor.
- Herausforderung: Globale Fairness, obwohl der Scheduler nur eine lokale Entscheidung über die Ressourcenvergabe treffen kann.
- Scheduling muss garantieren, dass Verbindungen, die mehr Daten senden als erlaubt, andere (faire) Verbindungen nicht beeinträchtigen (=***Absicherung***).

Max-min Fair Share

Formale Definition

- Ressourcen werden gemäß steigenden Anforderungen zugeteilt
- Keine Anforderung erhält mehr Ressourcen als benötigt
- Anforderungen, die ungenügend bedient werden, erhalten gleichen Anteil



Vorgehensweise

- Menge von Anforderungen $1, ..., n$ mit Ressourcenanforderungen $x_1, ..., x_n$. Die Anforderungen seien so angeordnet, dass gilt: $x_1 \leq x_2 \leq \ldots \leq x_n$
- Der Server habe eine Kapazität C. Dann erhält x_1 den Anteil C/n zugeteilt.
- Dies kann mehr sein als tatsächlich benötigt wird. In einem solchen Fall steht $C/n - x_1$ als ungenutzte Ressource weiter zur Verfügung und wird auf die weiteren Anforderungen gleichmäßig aufgeteilt, d.h. jede weitere Anforderung erhält (maximal) $C/n + (C/n - x_1)/(n-1)$.
- Dies setzt sich solange fort, bis eine Anforderung nicht erfüllt werden kann.

Ergebnis: Maximiert minimalen Anteil einer Anforderung, deren Forderung nicht erfüllt werden konnte

Verkehrsvertrag & Schedulable Region

- Leistungsschranken
  - Reservierung von Ressourcen, entweder on-the-fly (d.h. während des Verbindungsaufbaus) oder im Voraus
  - Zwischen Benutzer und Betreiber wird ein ***Verkehrsvertrag*** geschlossen, deterministische oder statistische Zusagen
  - Typische Leistungsparameter: Bandbreite, Verzögerung, Verzögerungsschwankung (Jitter) und Verluste
- Einfachheit und Effizienz der Zugangskontrolle
  - Die „Schedulabe Region“ ist die Menge möglicher Leistungsschranken, die ein Scheduler gleichzeitig einhalten kann, sie ist in der Regel nicht genau bekannt sondern wird approximiert.
  - Falls Schedulabe Region bekannt ist, ist die Zugangskontrolle einfach: Es muss lediglich geprüft werden, ob resultierende Kombination innerhalb der Schedulable Region liegt.

Vier Grundlegende Alternativen

Vier grundlegende Entwurfsentscheidungen sind zu treffen:

1. Wie viele Prioritätsstufen werden unterschieden?
2. Ist eine Stufe Work Conserving (=arbeitserhaltend) oder nicht?
    - Arbeitserhaltend heißt: der Scheduler ist nur dann im Leerlauf, wenn sich keine Anforderungen in den Warteschlangen befinden.
    - In der Praxis gilt heute nahezu immer Work Conserving.
3. Wie stark sollen Datenflüsse auf den Stufen aggregiert werden?
4. Welche Bedienreihenfolgen sollen innerhalb der Stufen gewählt werden?

1. Entscheidung: Wie viele Prioritätsstufen?

- Jedem Datenstrom wird eine Priorität zugeordnet
- Beispiel: Drei unterschiedliche Prioritätsklassen
  - Hohe Priorität für wichtige Dateneinheiten (z.B. zur Netzwerksteuerung)
  - Mittlere Priorität für garantierte Dienste
  - Niedrige Priorität für Best-Effort Dienste
- Problem
  - Absicherung der Datenströme ist nicht gegeben. Ein sich nicht konform verhaltender Datenstrom höherer Priorität beeinflusst diejenigen niedrigerer Priorität.
- Implementierung
  - Einfach, d.h. schnell ausführbar
  - Wenig Zustandshaltung erforderlich

2. Entscheidung: Arbeitserhaltend, ja/nein?

Falls ein Scheduler ***Work-Conserving*** (=arbeitserhaltend) ist, gilt:

- Die Summe der mittleren Warteschlangenverzögerungen einer Menge gemultiplexter Datenströme, gewichtet mit ihrem jeweiligen Anteil an der Last, ist *unabhängig* vom Scheduling-Algorithmus.

Formal

- N Datenströme, jeweils mit einer mittleren Datenrate von λ_i und einer mittleren Bedienzeit pro Dateneinheit von x_i.
- Die mittlere Auslastung des Übertragungsabschnitts durch Datenstrom i ergibt sich als: $\rho_i = \lambda_i x_i$
- Sei q_i die mittlere Wartezeit am Scheduler, dann besagt das Conservation Law das folgende:

$$\sum_{i=1}^{N} \rho_i q_i = \text{Constant}$$

Folge

- Scheduler kann die mittlere Warteschlangenverzögerung eines Datenstroms im Vergleich zum FIFO-Algorithmus nur zu Lasten anderer Datenströme reduzieren

3. Entscheidung: Aggregation, ja/nein?

- Aggregation
  - Einteilung der Datenströme in Klassen
  - Mehrere Datenströme werden aggregiert betrachtet hinsichtlich der Bedienreihenfolge
- Vorteil
  - Zustandshaltung reduziert sich
  - Somit: Skalierbarkeit auch für großen Netzen gewährleistet
- Nachteil
  - Datenströme einer Klasse sind nicht voneinander isoliert, d.h. ein sich nicht konform verhaltender Datenstrom beeinflusst die anderen Datenströme in seiner Klasse

4. Entscheidung: Bedienreihenfolge?

- Bedienreihenfolge für Best-Effort-Dienste
  - First In First Out (FIFO)
  - Generalized Processor Sharing (GPS)
  - Round Robin (RR)

- Bedienreihenfolge bei Dienstgarantien
  - Fair Queueing
  - Weighted Fair Queueing (WFQ)
  - Earliest Deadline First (EDF), auch Earliest Due Date (EDD) genannt

FIFO-Scheduling

- Nur eine Ausgangs-Warteschlange
- FIFO (First-In-First-Out) bzw. FCFS (First-Come-First-Served)
  - Dateneinheiten (bzw. Anforderungen) werden in Ankunftsreihenfolge bearbeitet
  - Ist kein Pufferplatz mehr frei, werden neu ankommende Dateneinheiten verworfen (engl. „Drop-Tail“)
- Vorteil: Sehr einfach zu implementieren
- Nachteile
  - Verantwortung für Staukontrolle an Endsystemen delegiert
  - Nicht möglich, einzelne Datenströme hinsichtlich Durchsatz oder Verzögerung zu bevorzugen
  - Datenströme mit langen Dateneinheiten erhalten besseren Service
  - "Gierige" Verbindungen beeinflussen andere
- Erweiterung auf Prioritäten möglich:
  - Warteschlangen mit unterschiedlichen Prioritäten.
  - Innerhalb einer Warteschlange: FIFO-Scheduling
  - Höchste Priorität wird stets zuerst bedient, d.h. keine definitive Zuordnung einer Verzögerung für einzelne Datenströme möglich
- Frage: Wie sollen die verschiedenen Warteschlangen bedient werden, um Fairness zu erreichen?
- Lösung: Konzept des ***Generalized Processor Sharing***

Earliest Deadline First

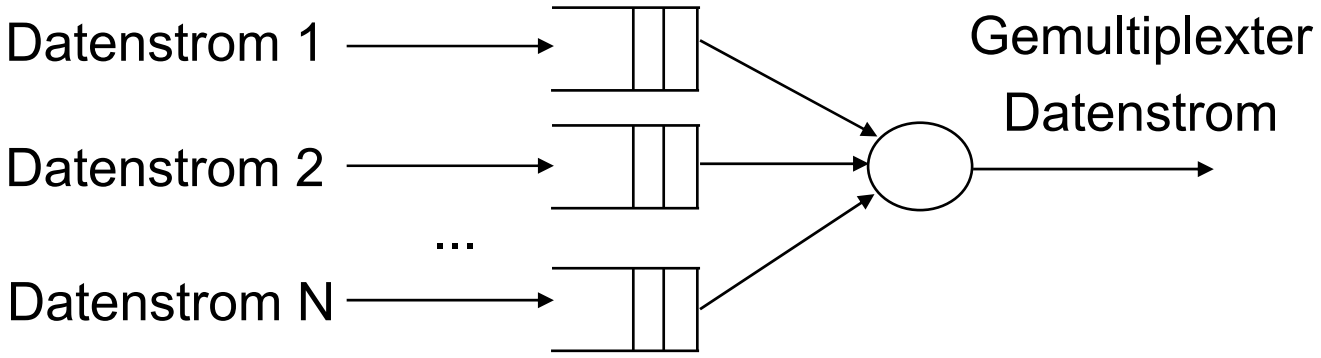
- Vorgehensweise
  - Jeder Dateneinheit wird eine Deadline zugewiesen. Scheduler bearbeitet Dateneinheiten gemäß deren Deadline.
  - Beim Verbindungsaufbau wird ein Verkehrsvertrag mit dem Scheduler ausgehandelt auf der Basis der maximalen Rate (Peak-Rate) einer Verbindung
    - Begrenzung der Worst-case-Verzögerung wird garantiert
    - Nachteil: Ressourcenreservierung auf der Basis der Peak-Rate führt zu schlechter Ausnutzung der Bandbreite
- Vorteil
  - Garantie der Ende-zu-Ende-Verzögerung unabhängig von der Bandbreite
- Nachteil
  - Aufwändige Implementierung (Zeitstempel, Sortierung)

Generalized Processor Sharing

- Ziel bei Best-Effort-Diensten ist „Max-min Fair Share“ (= Definition von „Fairness“) zu erreichen.
- Generalized Processor Sharing:
  - Ideales Work Conserving Scheduling, d.h. Restbandbreite wird fair zwischen konkurrierenden Datenströme aufgeteilt
  - Theoretisches Konzept, d.h. nicht praktisch implementierbar
- GPS bildet die theoretische Grundlage für die Beurteilung praktisch implementierbarer Scheduling-Verfahren
- Vorgehen
  - Dateneinheiten werden pro Datenstrom unterschiedlichen Warteschlangen zugeordnet
  - Jede nicht leere Warteschlange wird regelmäßig besucht und eine unendlich kleine Menge der Dateneinheit bedient (unmöglich in der Praxis)
  - In jeder endlichen Zeit können damit alle Warteschlangen besucht werden
- Formal
  - Datenstrom i erhält ein Gewicht $\phi(i)$
  - Server bedient $S(i, \tau, t)$ Daten des i-ten Datenstroms im Intervall $[\tau, t]$
  - Für alle anderen Datenströme j, die ebenfalls Daten zu bedienen haben, gilt: $\frac{S(i,\tau,t)}{S(j,\tau,t)} \geq \frac{\phi(i)}{\phi(j)}$

Round-Robin-Scheduling

- Vorgehen
  - Bedient in einer Runde eine Dateneinheit jeder nicht leeren Warteschlange („Reihum-Verfahren“)
  - Emuliert GPS gut, wenn alle Datenströme gleiche Gewichte haben und die Dateneinheiten alle gleich lang sind



Weighted Round-Robin

- Berücksichtigt Gewichte der Datenströme
- Problem
  - Unfair, wenn Pakete unterschiedliche Länge besitzen
  - D.h. bei verschieden langen Dateneinheiten muss mittlere Länge bekannt sein, um GPS gut zu approximieren
    - Normalisiert das Gewicht eines Datenstroms mit der mittleren Größe der Dateneinheiten, z.B. Gewichte (0.5,0.75,1.0), mittlere Paketgrößen (50,100,1500) → Gewichtung (60,45,4)
  - Fairness ist nur für Zeithorizonte gegeben, die länger als eine Runde sind
  - D.h. bei kleinen Gewichten oder großer Anzahl von Datenströmen kann dies zu langem Zeitraum der Unfairness führen
- Vorteil
  - Verkehr einer unfairen Quelle belastet andere Datenströme nicht

Fair Queuing

Approximation eines „Bit-für-Bit“-Round-Robin

– Ermitteln des Zeitpunktes, zu dem das Senden einer Dateneinheit mit einem Bit-für-Bit Round-Robin theoretisch beendet wäre (*Endezeit*)

– Pakete werden dann in Reihenfolge ihrer virtuellen Endezeit bedient

Praktische Vorgehensweise:

– Eine *Rundenzahl* wird bestimmt, welche die Anzahl der Bit-für-Bit-Runden angibt, die zu einem gegebenen Zeitpunkt beendet sind

– Jede Runde nimmt eine variable (reale) Zeitspanne in Anspruch

– Trifft Paket der Länge D auf

  • leere Warteschlange (inaktiver Datenstrom) bei Rundenzahl R, so ist seine Endezeit $R+D$

  • volle Warteschlange (aktiver Datenstrom), so ist seine Endezeit $F+D$, wenn F die Rundenzahl des vorhergehenden Pakets ist

– Ist die Rundenzahl bekannt, so kann die Endezeit also folgendermaßen berechnet werden

  • $D(i,k,t)$ Länge der k-ten Dateneinheit, die zur Zeit t beim Datenstrom i ankommt

  • $R(t)$ sei die Rundenzahl zum Zeitpunkt t

  • $F(i,k-1,t)$ sei Endezeit der $k-1$-ten Dateneinheit des Datenstroms i zum Zeitpunkt t

  • $F(i,k,t) = max\{ F(i,k-1,t), R(t) \} + D(i,k,t)$

– Definition der Rundenzahl als Variable, die invers zur Anzahl aktiver Datenströme wächst, d.h. $R(t)$ wächst mit zunehmender Anzahl aktiver Datenströme langsamer

Weighted Fair Queueing

- Problem: Berechnung der aktuellen Rundenzahl ist schwierig
  - Abhängig von Datenströmen die inaktiv werden und die eventuell andere inaktive Datenströme nach sich ziehen
  - Bei jeder Änderung (Hinzunahme/Wegfall von Datenströmen) ist eine komplette Neuberechnung erforderlich, d.h. Zustandshaltung pro Datenstrom erforderlich
- Berechnung der Endezeit:

$$F(i,k,t) = \max\{F(i,k-1,t),\ R(t)\} + \frac{D(i,k,t)}{\phi(i)}$$

- Rundenzahl erhöht sich mit $(\Sigma_{i=1}^{N(t)} \Phi(i))^{-1}$
  wobei $N(t)$ ist die zum Zeitpunkt t aktive Anzahl an Datenströmen
- Positive Eigenschaften
  - Datenströme sind voneinander abgesichert
  - Angabe von oberer Schranke für Ende-zu-Ende-Verzögerung möglich
  - Benutzer sollten Flusskontrolle implementieren, ansonsten Verluste von Dateneinheiten möglich

Beispiel zu WFQ

Aufgabe

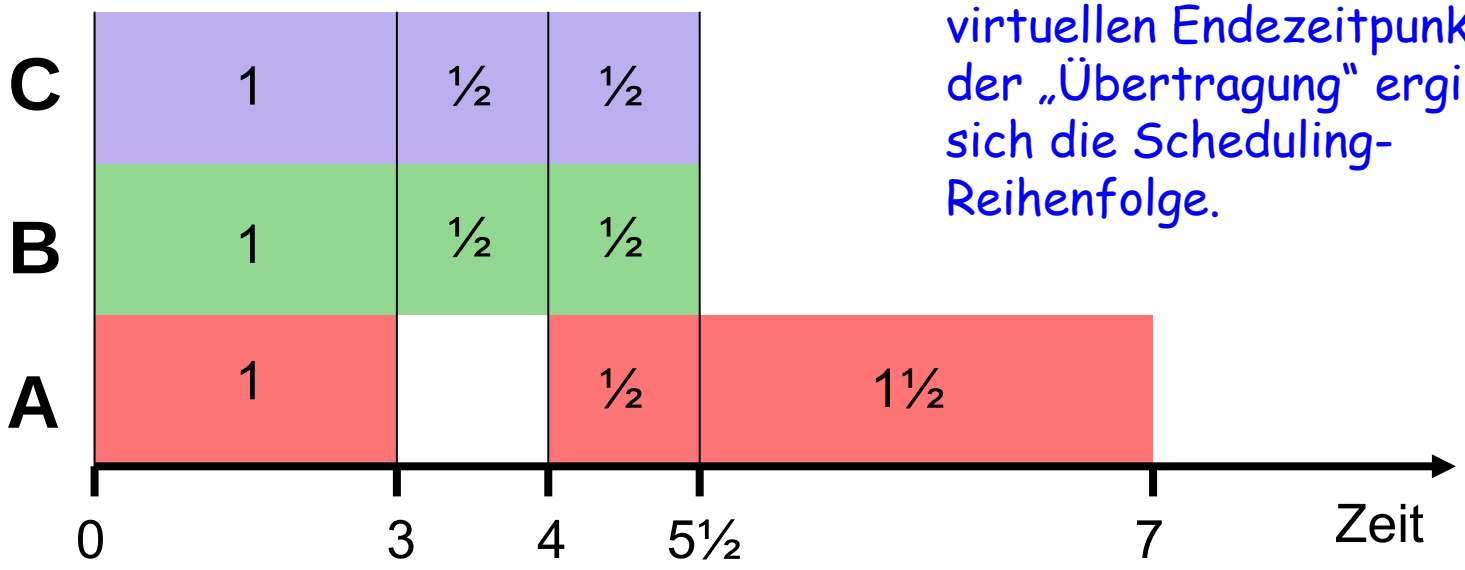
- Dateneinheiten der Größen 1, 2, und 2 erreichen einen WFQ-Scheduler zum Zeitpunkt 0. Sie gehören zu den gleichgewichteten Datenströmen A, B und C. Für Datenstrom A wird zum Zeitpunkt 4 eine weitere Dateneinheit der Länge 2 empfangen. Die Datenrate sei eine Größeneinheit/Sekunde.
- Bestimmen Sie die Endezeiten der Dateneinheiten. Bei welcher Rundenzahl befindet sich das System im Leerlauf?

Aus der Berechnungsformel $F(i,k,t) = \max\{ F(i,k-1,t), R(t) \} + D(i,k,t)$ folgt:

- $F(A,0,0) = 0 + 1 = 1$
- $F(B,0,0) = 0 + 2 = 2$
- $F(C,0,0) = 0 + 2 = 2$
- $F(A,1,4) = \max\{1, 1.5\} + 2 = 3.5$

Das kann man sich wie folgt veranschaulichen:
→ nächste Folien

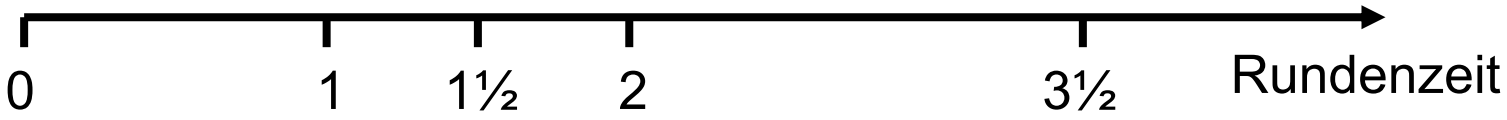
Beispiel zu WFQ – Lösung (1)



Aus der Reihenfolge der virtuellen Endezeitpunkte der „Übertragung" ergibt sich die Scheduling-Reihenfolge.

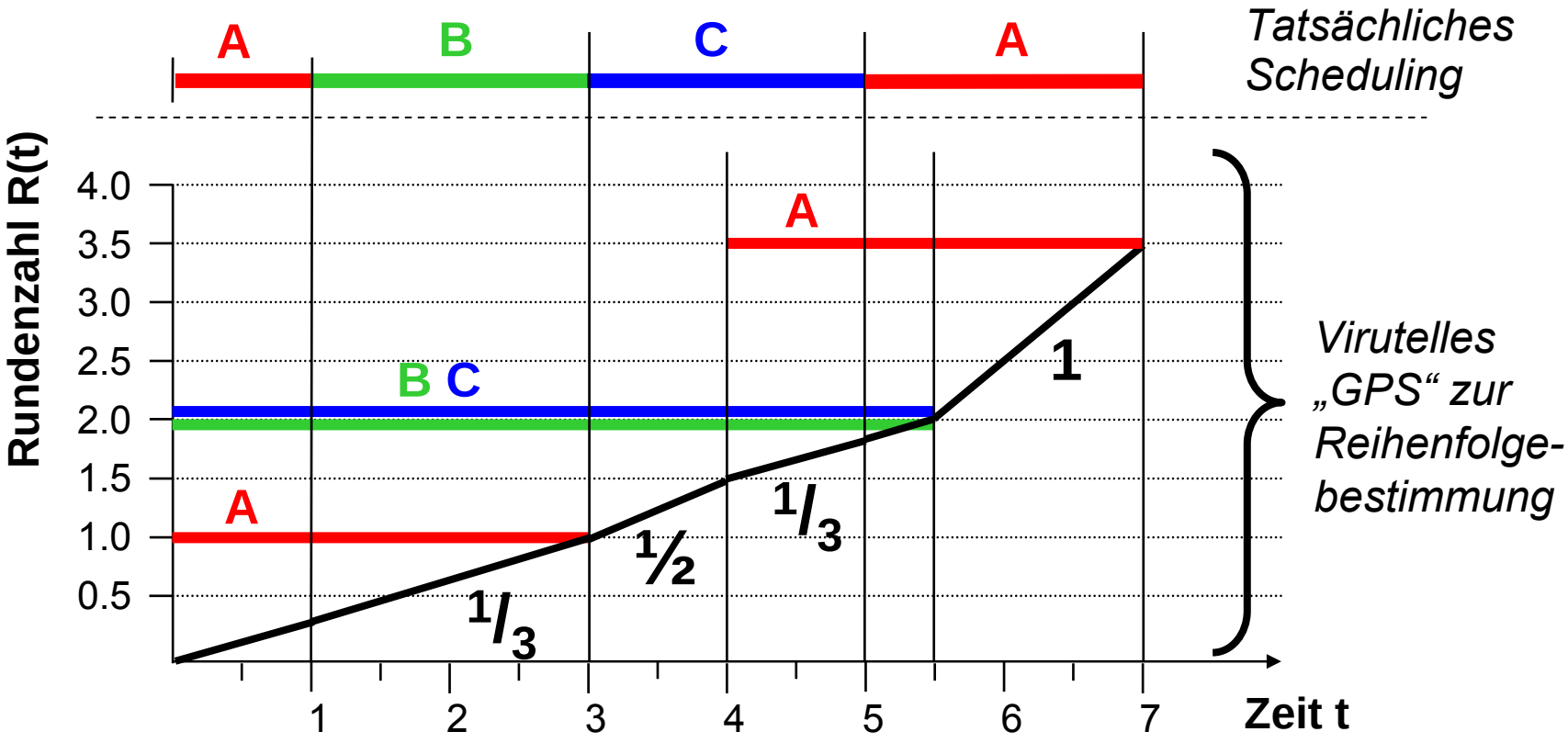
Achtung: *Die obige Zeitachse ist nicht maßstabsgetreu, denn je mehr Warteschlangen gefüllt sind, desto länger dauert die „Übertragung" einer bestimmten Datenmenge!*

Deshalb: *Umskalieren zur „Rundenzeit", im Bild einfaches Aufaddieren ohne Beachtung der Zahl gefüllter Warteschlangen:*



Beispiel zu WFQ – Lösung (2)

Diesen Zusammenhang zwischen Rundenzeit und tatsächlicher Zeit kann man auch wie folgt darstellen:



WFQ – Noch ein Beispiel

- Datenströme A, B und C haben folgende Dateneinheiten zu verschicken: D(A,0,0)=2, D(B,0,0)=2, D(C,0,1)=1, D(C,1,2)=1
- Das heißt: Datenstrom A und B haben jeweils zum Zeitpunkt 0 eine Dateneinheit der Größeneinheit 2 zu senden, Datenstrom C habe zu den Zeitpunkten 1 und 2 jeweils eine Dateneinheit der Größe 1 zu senden. Datenstrom C habe außerdem die Gewichtung $\phi(C)=2$, Datenströme A und B jeweils $\phi(i)=1$. Ausgangsdatenrate: eine Größeneinheit/s.

Noch zu sendende Datenmenge	A	2	1.5	1.25	1	0.75	0.5	0
	B	2	1.5	1.25	1	0.75	0.5	0
	C	0	1	1.5	1	0.5	0	0
	Reale Zeit t[s]	0	1	2	3	4	5	6
	Rundenzeit R(t)	0	0.5	0.75	1	1.25	1.5	2

- F(A,0,0)= 2, F(B,0,0)= 2, F(C,0,1)= max(0,R(1))+1/2= max(0,0.5)+1/2=1
- F(C,1,2)= max(0,R(2))+1/2= max(1,0.75)+1/2=1.5
- Nach 6s ist der Scheduler fertig und die durch die berechneten Endezeitpunkte bestimmte Sendereihenfolge sieht folgendermaßen aus: A,C0,C1,B (alternativ: B,C0,C1,A)

WFQ – Verzögerungsgarantie (1)

Weighted Fair Queuing Scheduling erlaubt es, eine obere Schranke für die maximale Ende-zu-Ende-Verzögerung zu berechnen.

- Voraussetzung: Quelle *i* kann während der Zeitspanne *Δt* nur *σ(i)* + *ρ(i) Δt* Bits senden, wobei *σ* die Tiefe (d.h. max. Burstgröße) und *ρ* die Rate eines Token-Bucket sind.
- Beobachtung: WFQ limitiert die den einzelnen Datenströmen zugeordnete Bandbreite, da dem Datenstrom i beim h-ten Hop bzw. Scheduler der folgende Anteil der Bandbreite zugeordnet wird:

$$\frac{\phi(i,h)}{\sum_{j=1}^{Nh}\phi(j,h)}$$

wobei N_h die Anzahl der Datenströme im Scheduler h ist

WFQ – Verzögerungsgarantie (2)

– Für die Bedienrate $g(i,h)$, die einem Datenstrom i vom h-ten durchlaufenen Scheduler zugewiesen wird, gilt dann:

$$g(i,h) = r(h) \cdot \frac{\phi(i,h)}{\sum_{j=1}^{N_h} \phi(j,h)}$$

wobei $r(h)$ die Datenrate des Übertragungsabschnitts (=Link) ist

– Annahme: Die minimal garantierte Datenrate $g(i)$ entlang des Wegs sei größer als die Senderate, d.h. $g(i) := min_{h=1..H}\, g(i,h) \geq \rho\,(i)$

– Bemerkung: Ohne diese Annahme stiege ja die Warteschlange an einem Scheduler ohne Grenze!

– Definieren wir weiter:

- L_{max} sei die größte erlaubte Dateneinheit (auch für Pakete anderer Ströme!)
- H die Anzahl der Scheduler

WFQ – Verzögerungsgarantie (3)

Obere Schranke für die Verzögerung eines Pakets des Datenstroms i ist dann:

$$\frac{\sigma(i)}{g(i)} + \sum_{h=1}^{H}\left(\frac{L_{max}}{r(h)} + \frac{L_{max}}{g(i,h)}\right)$$

$\frac{\sigma(i)}{g(i)}$ Bedienzeit für max. Burst mit minimal garantierter Rate

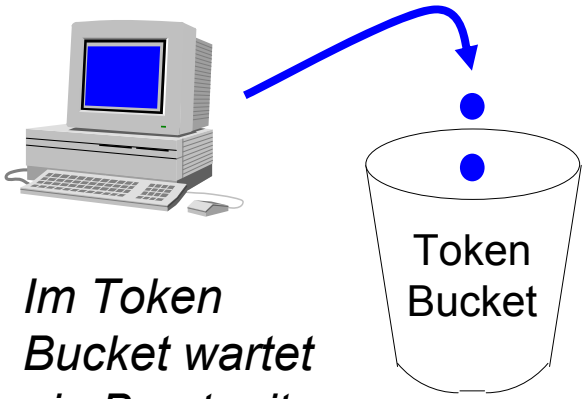
$\frac{L_{max}}{g(i,h)}$ Bedienzeit für max. langes Paket mit zugesicherter Rate

$\frac{L_{max}}{r(h)}$ Übertragungsdauer eines Pakets mit max. Länge bei Linkspeed (nicht verdrängbares Paket das gerade zuvor bedient wird)

WFQ – Verzögerungsgarantie (4)

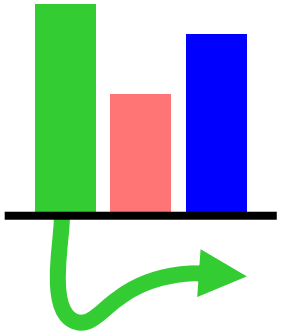
... und das ganze noch mal anschaulich:

$$\frac{\sigma(i)}{g(i)} + \sum_{h=1}^{H}\left(\frac{L_{max}}{r(h)} + \frac{L_{max}}{g(i,h)}\right)$$



Im Token Bucket wartet ein Burst mit Größe σ, der mit der Rate g vom System bedient wird.

Kurz vor Ankunft des eigenen (blauen) Pakets wurde angefangen ein grünes Paket zu übertragen. (GPS mit seinen infinitesimalen Bedienzeiten schließt diesen Fall aus!)



Danach erst kann das reguläre Scheduling mit der Bedienrate g beginnen.

The Magic Cure: Over-Provisioning

- Observation: Everything discussed about quality of service, today, comes only into effect when there are not enough resources to satisfy the demand entirely.
- Conclusion: Just provide enough resources for all!
- Multimedia in the Internet has become widely used, although QoS protocols are not used in practice.

Ressourcenverwaltung

Bereitstellung von Dienstgüte erfordert

- Betriebsmittelverwaltung (engl. Resource Management).
- Ohne Betriebsmittel und deren Verwaltung keine Dienstgüte!

Zwei Verfahren zur Bereitstellung von „echter“ QoS:

- Ressourcenreservierung: Ressourcen (= Speicher, CPU, Bandbreite) werden vor ihrer Benutzung reserviert.
- Skalierung: Anpassung des Datenflusses an gegebene Randbedingungen. Dabei interagieren die Betriebsmittel mit dem Datenstrom.

Zusätzlich im „klassischen“ Internet:

- Overprovisioning: Es werden mehr Ressourcen installiert als voraussichtlich benötigt werden.
- Endgeräte-Fairness (vgl. TCP Fairness): Endgeräte erkennen Ressourcen-Engpässe und reduzieren ihre Anforderungen ans Netz.

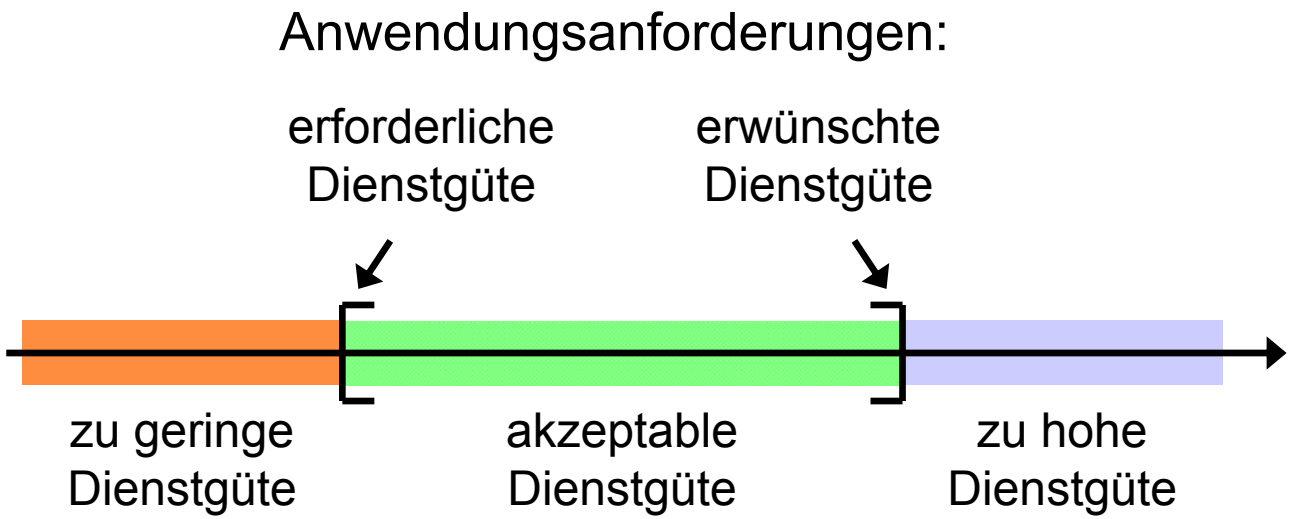
Typische Dienstgüte-Parameter

- Verzögerung
  - Maximale Ende-zu-Ende Verzögerung für ein Paket
  - Maximale Varianz (Jitter) für Übertragungszeiten
- Durchsatz
  - Maximale Anzahl von Dateneinheiten pro Zeitintervall
  - Maximale Paket Größe
- Verlust
  - Prioritätsklasse (wichtig bzw. unwichtig)
  - Verlustrate: Maximal erlaubte Anzahl von verlorenen Paketen pro Zeiteinheit
  - Verlustgröße: Maximal erlaubte Anzahl von aufeinanderfolgend verlorenen Paketen

Dienstgüte-Spezifikation

- Dienstgüte-Spezifikation dient zur
  - Spezifikation der Anforderungen der Anwendung, zur
  - Beschreibung der vom System garantierte Dienstgüte und zur
  - Beschreibung des Status' von Betriebmittelreservierungen.
- Dienstgüte-Spezifikation beinhaltet
  - Dienstgüteparameter: Notwendige und gewünschte Dienstgüte
  - Dienstklasse (Diensttyp): Zusammenfassung von Dienstgüteparametern
- Spezifikation der Anwendungsanforderungen durch einzelne Werte für die Dienstgüte-Parameter ist unrealistisch; deshalb Spezifikation eines Intervalls zwischen
  - Erforderliche Dienstgüte, d.h. mindestens notwendige Dienstgüte damit die Anwendung überhaupt funktioniert, und
  - Erwünschter Dienstgüte, d.h. maximal ‚sinnvolle' Dienstgüte.

Dienstgüte-Intervall

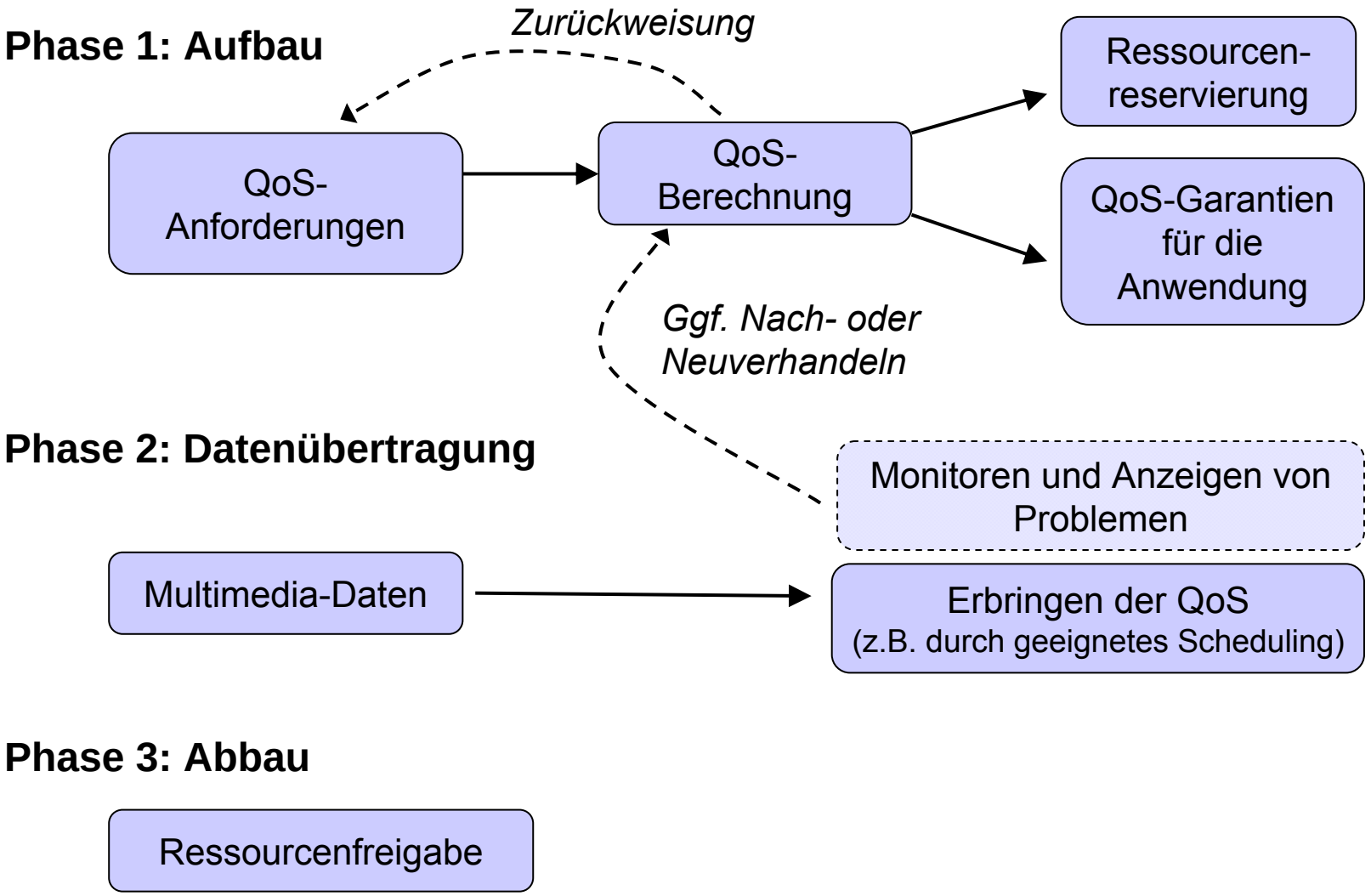


Dienstklassen

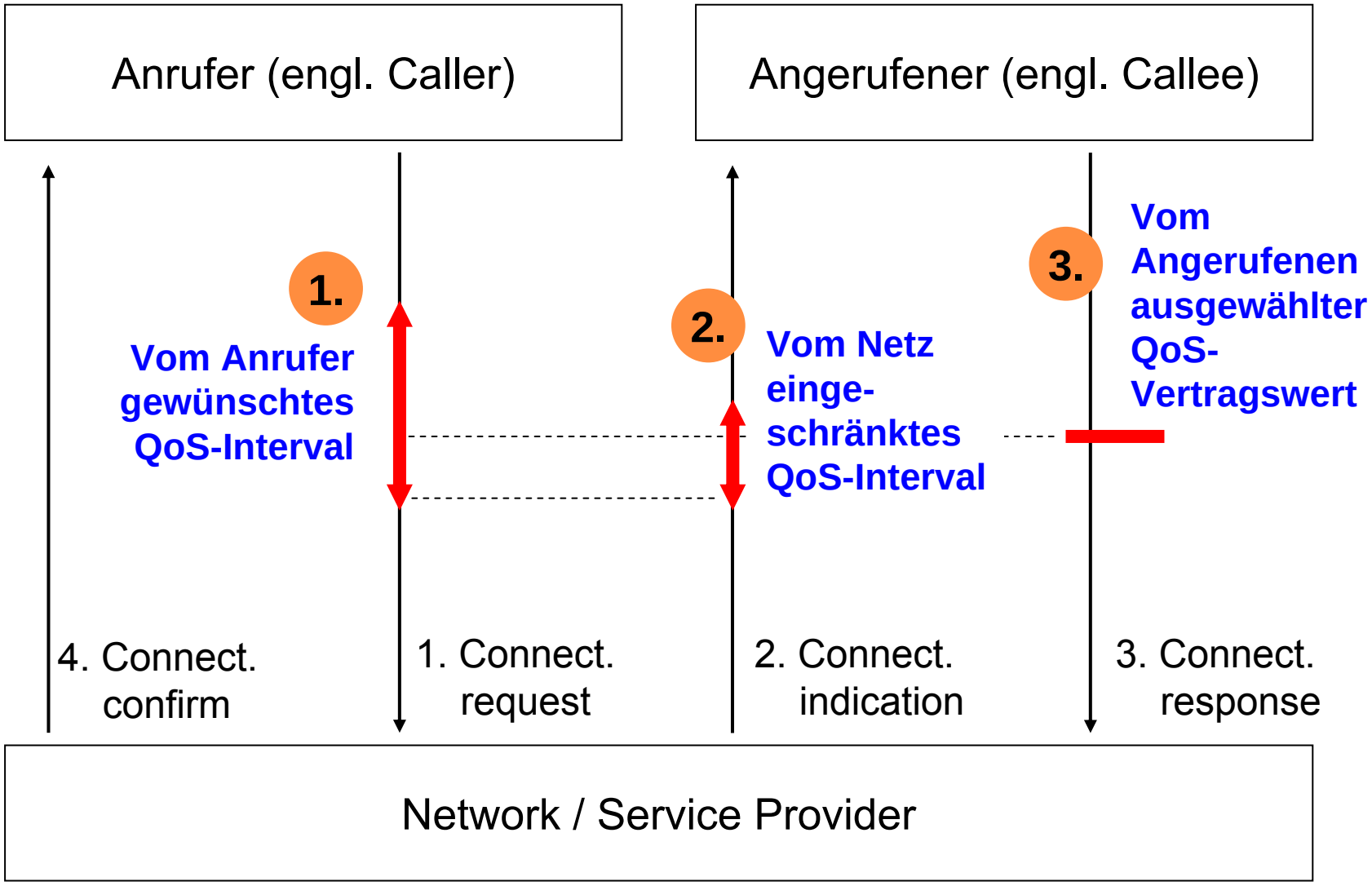
Verbindlichkeit der Dienstgüte-Parameterwerte wird durch Dienstklasse (Diensttyp) beschrieben:

- **Deterministisch garantierte Dienste**
  - Dienstgüte wird durch einen konkreten Wert als harte (obere) Schranke spezifiziert.
  - Zuteilung basiert auf Wort-Case-Annahmen für das Systemverhalten.
  - Nachteil: Schlechte Ressourcennutzung durch „Überreservierung“.
- **Statistisch garantierte Dienste**
  - Dienstgüte wird durch einen statistischen Wert spezifiziert und muss nur „im Mittel“ bzw. mit einer bestimmten Wahrscheinlichkeit eingehalten werden.
  - Zuteilung basiert auf einem stochastischen Modell des Systemverhaltens.
- **Vorhersagbare Dienste**
  - Es gibt nur Aussagen, wie die Dienstgüte in der Vergangenheit war.
  - Eigenschaften sind ähnlich den statistisch garantierten Diensten; die Zuteilung ist einfacher, weil kein Modell benötigt wird.
- **Best-Effort Dienste**
  - Keine Dienstgütegarantien, sondern die Hoffnung, dass die Ressourcen und die Fairness der Endgeräte bzw. Anwender ausreichen.

Phasen der Betriebsmittelverwaltung



Trilaterale Dienstgüteverhandlung



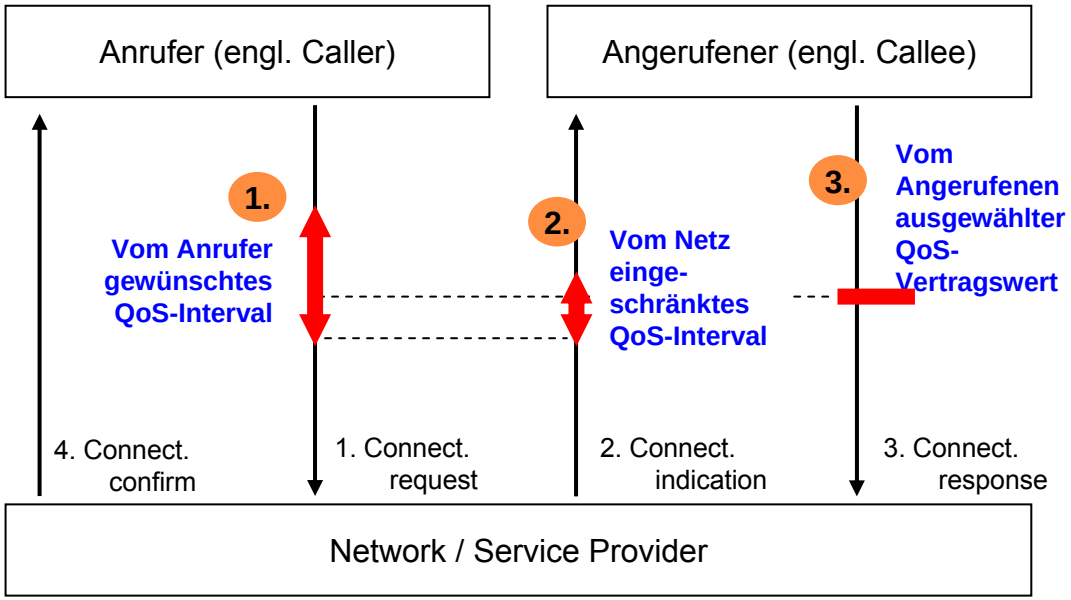
Dienstgüte im Internet

Es gab bereits mehrere Ansätze für Dienstgüte im Internet:

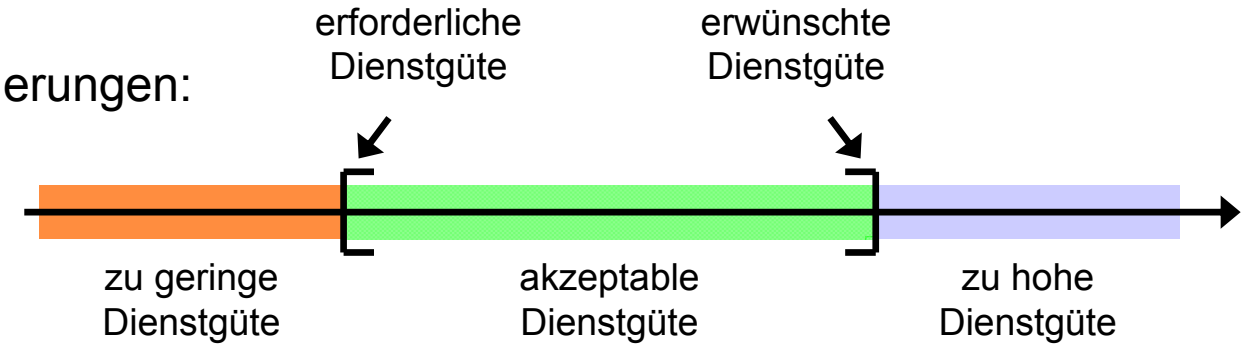
- IP ***Type of Service***, hat sich nicht durchgesetzt
- ***Stream Transport Protocol Version 2+*** (ST2+):
  - Hard-State für Reservierungsinformation, eigenes Protokoll neben IP
  - Nicht in der Praxis eingesetzt, nur noch von historischem Interesse
- ***Integrated Services***
  - Drei festgelegte Dienste: Guaranteed Service, Controlled Load, Best-Effort
  - Empfängerorientiertes Signalisierungsprotokoll RSVP
  - Soft-State für Reservierungsinformation
  - Zu aufwendig für die Praxis, hat sich nicht durchgesetzt
- ***Differentiated Services***
  - Unterstützung für max. 64 verschiedene Dienstklassen
  - Implizite Signalisierung der Dienstklasse in der Dateneinheit
  - Aggregation von Verkehr reduziert Aufwand im Netzinneren
  - Hat sich (noch) nicht durchgesetzt …

Dienstgüte

- *Anwendungen benötigen ggf. Garantien für minimale Bandbreite, maximale Verzögerung, max. Jitter, etc.*
- *Diese Dienstgüteparameter müssen mit dem Netz und der Gegenstelle ausgehandelt werden.*



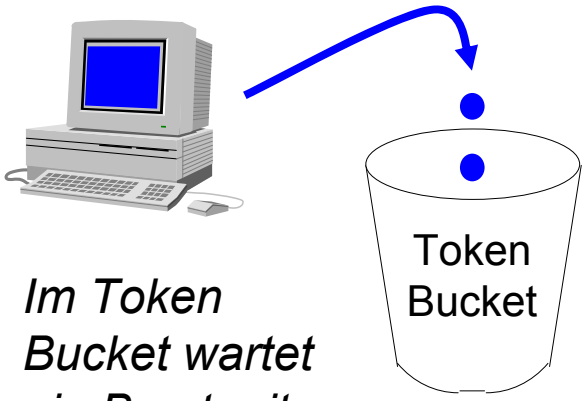
Anwendungsanforderungen:



Weighted Fair Queueing

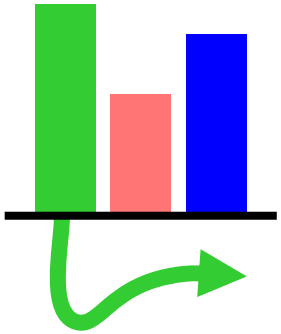
WFQ kann obere Schranken der Paket-laufzeit garantieren.

$$\frac{\sigma(i)}{g(i)} + \sum_{h=1}^{H}\left(\frac{L_{max}}{r(h)} + \frac{L_{max}}{g(i,h)}\right)$$



Im Token Bucket wartet ein Burst mit Größe σ, der mit der Rate g vom System bedient wird.

Kurz vor Ankunft des eigenen (blauen) Pakets wurde angefangen ein grünes Paket zu übertragen. (GPS mit seinen infinitesimalen Bedienzeiten schließt diesen Fall aus!)



Danach erst kann das reguläre Scheduling mit der Bedienrate g beginnen.

Questions?

Thomas Fuhrmann

Department of Informatics
Self-Organizing Systems Group
c/o I8 Network Architectures and Services
Technical University Munich, Germany

fuhrmann@net.in.tum.de