Faculty of Computer Science

universität wien

Netzwerktechnologie für Multimedia Anwendungen (NTM)

Kapitel 4

Florian Metzger
florian.metzger@univie.ac.at

David Stezenbach
david.stezenbach@univie.ac.at

Bachelorstudium Informatik
WS 2014/2015

Mit Material von

"Controlling Queue Delay"; K. Nichols; http://queue.acm.org/detail.cfm?id=2209336
http://www.bufferbloat.net/attachments/download/147/Not_every_packet_is_sacred-Fixing_Bufferbloat_Codel_Fq_Codel.pdf
Inside Codel and Fq Codel, D. That, http://mirrors.bufferbloat.net/Talks/Stanford2013

4. Quality of Service

- 4.1 Grundlagen
- 4.1 Warteschlangen Management
- 4.2 Data Plane QoS Mechanismen
- 4.3 Control Plane QoS Mechanismen

4. Quality of Service

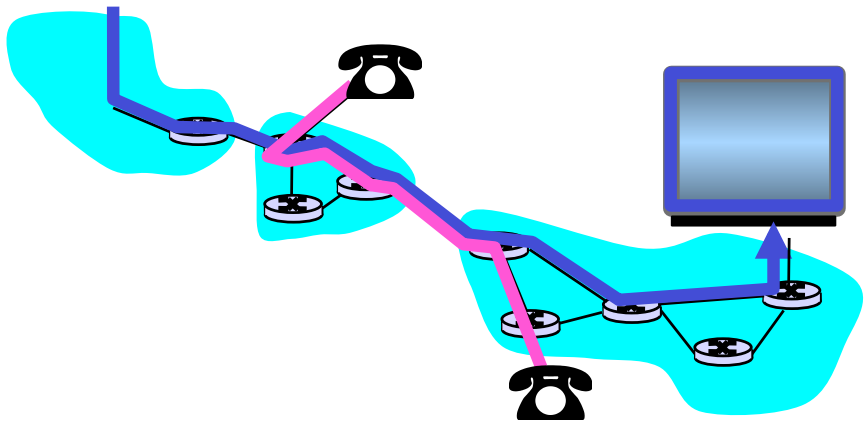
- **4.1 Grundlagen**
- 4.1 Warteschlangen Management
- 4.2 Data Plane QoS Mechanismen
- 4.3 Control Plane QoS Mechanismen

Was ist Quality of Service (QoS)?

- Generell:
  - Über ein Netzwerk werden von Multimedia-Anwendungen Daten übertragen
    (Audio- und Videoübertragung → “kontinuierliche Medien”)
  - Netzwerke bieten dafür (Transport)-Dienste an

QoS

Das Netzwerk soll das Niveau der **Dienstgüte** anbieten, das für die jeweilige Anwendung notwendig ist.



Definition – Quality of Service

Eine Beispieldefinition von CISCO:

> ***QoS** refers to the **ability of a network** to provide **improved service to selected network traffic** over various underlying technologies including Frame Relay, ATM, Ethernet and 802.1 networks, SONET, and IP-routed networks.*

Eine Beispieldefinition von ITU-T (1994):

> ***Quality of service (QoS):** Totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied **needs of the user of the service.***

Definition laut Wikipedia

> **Quality of service (QoS)** is the overall performance of a telephony or computer network, particularly the performance seen by the users of the network.To **quantitatively measure quality of service**, several related aspects of the network service are often considered, such as **error rates, bandwidth, throughput, transmission delay, availability, jitter,** etc.

Netzwerk-QoS Metriken

- *Bottleneck Bandwidth*
- *Throughput*
- *Network Latency / Delay*
- *Jitter*
- *Packet Loss*
- *Per Flow Sequence Preservation*
- *Availability, Reliability*

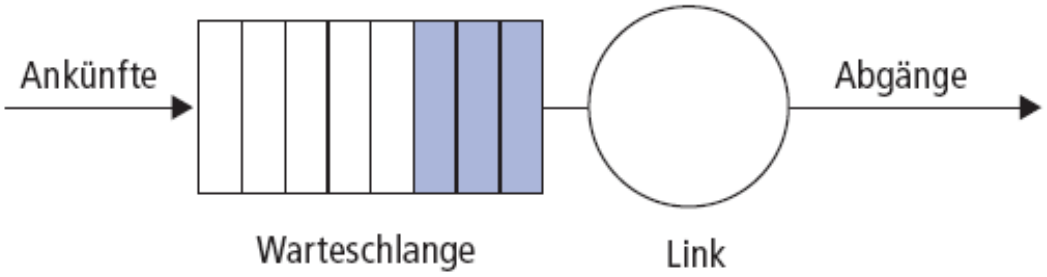
4. Quality of Service

- 4.1 Grundlagen
- **4.1 Warteschlangen Management**
- 4.2 Data Plane QoS Mechanismen
- 4.3 Control Plane QoS Mechanismen

FIFO Scheduling

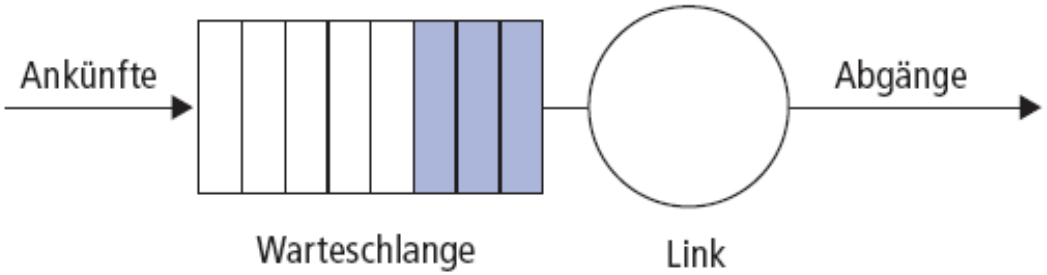
First-In First-Out (FIFO) = First-Come First-Served (FCFS)

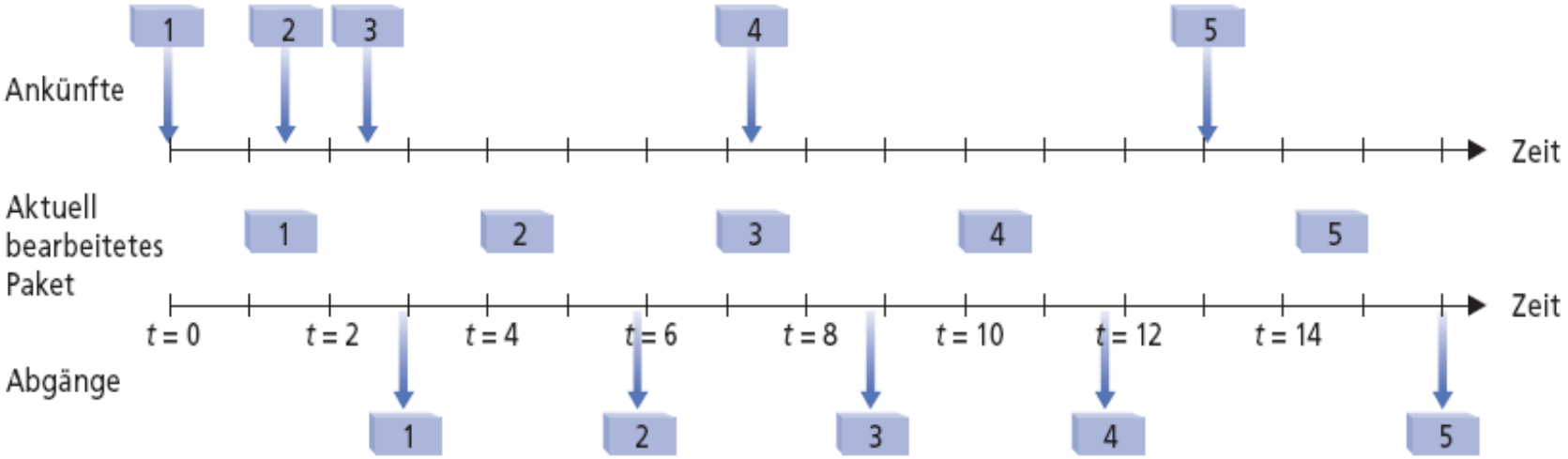
- Bearbeitung in der Reihenfolge der Ankunft



- Die Warteschlange wird **aktiv**, sobald mindestens ein Paket darin liegt
- Durch **Klassifizierung** können **mehrere Warteschlangen** entstehen

Arbeitsweise einer FIFO-Warteschlange





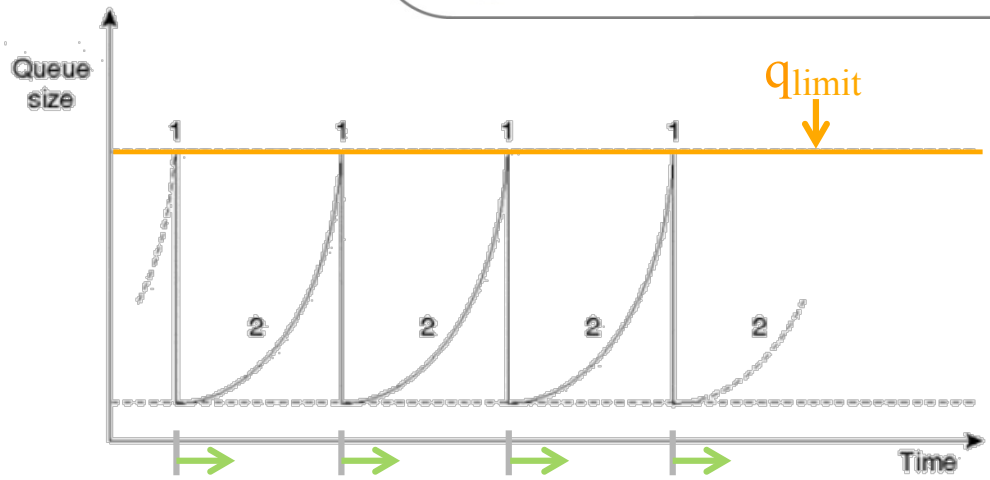
Tail Drop / Drop Tail

- Ein Queue Management Algorithmus der von Internet Routern verwendet wird um festzustellen wann Pakete verworfen werden sollen
- Wenn die Warteschlange voll ist werden weitere ankommende Pakete verworfen bis in der Warteschlange wieder Platz ist
- Jedes Paket wird gleich behandelt

Verwerfung der Pakete (*Packet Dropping*)

- Router Line Card Speicher
  - Puffer: gemeinsamer Speicher, in dem alle zu sendenden Pakete landen
  - Warteschlange: Liste, die auf Pakete im Puffer zeigt
- Wenn *Ankunftsrate* > *Servicerate*, dann:
  - Puffer füllt sich an, wird irgendwann bis zur (vor)gegebenen Beschränkung voll
  - Pakete werden verworfen (*Packet Dropping*)
- Wichtiger Grund für restriktive Warteschlangengrößen bzw. für *Packet-Dropping*:
  - Limitierung der dadurch entstehenden Verzögerungen!

Tail Drop - Problem



Wenn mehrere TCP Verbindungen bestehen kann es vorkommen, dass Pakete mehrerer TCP Flows verworfen werden (wenn der Puffer voll ist).

1. Die TCP-Sender erkennen, dass Pakete verloren gegangen sind und treten gleichzeitig in die Slow-Start-Phase ein in der sie ihre Senderate auf ein Minimum reduzieren.
   → *Der Netzwerk-Durchsatz sinkt schlagartig!*
   → *Das Netzwerk kann nun **unterbelastet** sein.*

2. Wenn die TCP-Sender wieder *Acknowledgements* vom Empfänger erhalten vergrößern sie ihr *Congestion Window* laufend.
   → *Alle TCP-Sender erhöhen schrittweise ihre Senderate bis der Buffer wieder voll ist!*
   → *Das Netzwerk ist **verstopft**.*

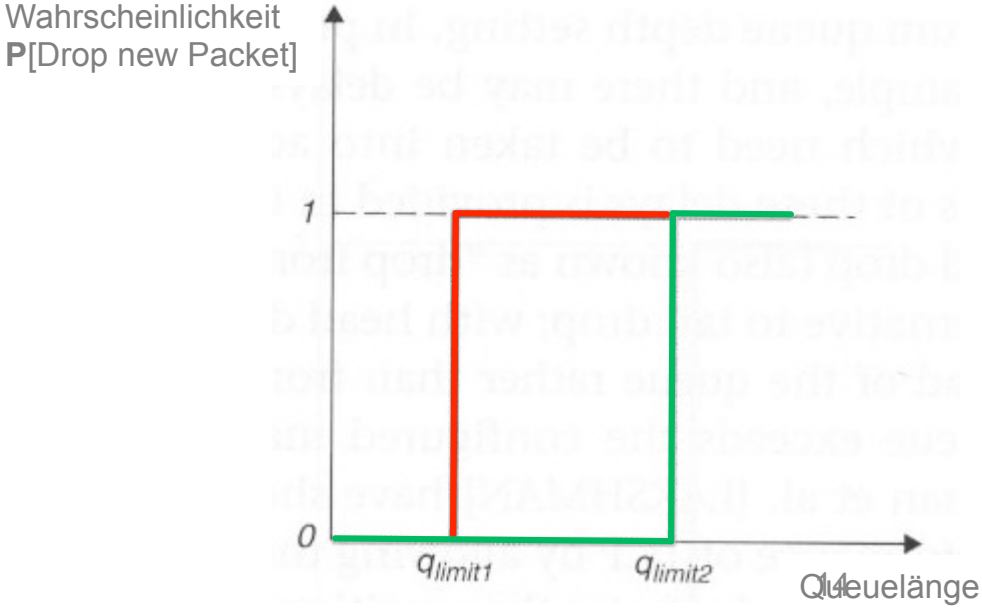
Diagramm angelehnt an http://ivanyu.files.wordpress.com/2011/04/15.gif

Weighted Tail Drop

- **Zwei Typen** von Paketen (z.B. von einem *Policer* markiert)
  - *Out-of-contract* Paket wird nur behalten wenn Queuelänge < q_{limit1}
  - *In-contract* Paket wird behalten solange Queuelänge < q_{limit2}

Beispiel: Es seien *N* Pakete in der Queue mit $q_{limit1} < N < q_{limit2}$

- Ein *out-of-contract Paket* würde **verworfen** werden
- Ein *in-contract Paket* würde **eingefügt** werden



Random Early Detection (RED)

- Ein Active Queue Management und Congestion Avoidance Algorithmus

- Bei Tail Drop:
  - Platz im Puffer wird nicht fair auf die Flows aufgeteilt
  - Bei mehreren TCP Verbindungen wird das Netzwerk im Wechsel überflutet und unterbelastet

- RED versucht dies zu berücksichtigen:
  - Einzelne Sessions sollen früher als andere zurückschalten
  - RED überwacht dazu die durchschnittliche Queuelänge qavg und berechnet die davon abhängende Wahrscheinlichkeit P[Drop Packet]

Probleme Klassischer AQM

- RED (Blue, etc) - 1992-2002
  - Verwalten Puffer durch Abschätzen einer guten Größe durch diverse Algorithmen
  - Droppen/markieren zufällige Pakete nach einer Menge von Regeln und Bedingungen
  - Funktionieren nur eingeschränkt bei hohen Bandbreiten
  - Sehr schwer zu konfigurieren und nur für jeweils genau eine bestimmte Bandbreite
  - Unterscheiden nicht zwischen verschiedenen Flows
  - Wird nicht eingesetzt
  - Nur eingeschränkte Abmilderung von „Latency Spikes“

Nach 2002

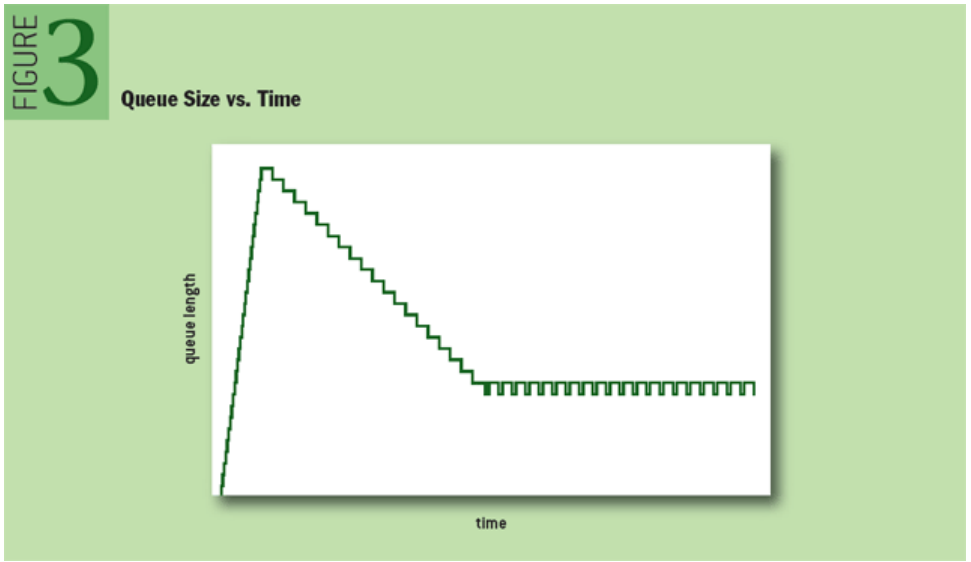
- Kaum Queue Management Forschung
  - Real eingesetzte Traffic Shaping Anwendungen und Verbesserungen (z.B. “wondershaper” (2002) und adsl-shaper (2005)) wurden nicht weiter untersucht und ignoriert
- Mechanismen wurden höchstens zum QoS-Priorisierung und Verkehrsdifferenzierung angewandt
  - Nicht für „fair queuing“ von Flows
  - Kein Einsatz von Packet Drop
- Netzwerke wurden sehr viel schneller
  - Daher veränderte Warteschlangen und Paketverzögerungen
  - Drahtlosnetze bringen häufig noch mehr Puffer mit sich

Bufferbloat

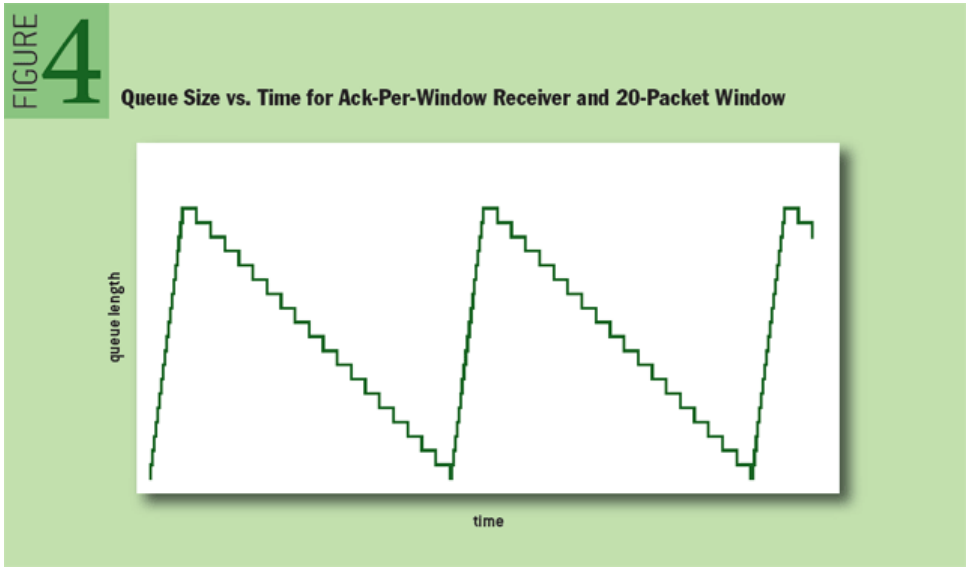
- Definitionen und Ursachen
  - “A phenomenon in a packet-switched computer network whereby excess buffering of packets inside the network causes high latency and jitter, as well as reducing the overall network throughput.” (Wikipedia)
  - Übermäßiges Buffering an der Netzwerkkarte, im Treiber, in der Netzwerkwarteschlange, in den TCP/UDP Queues, … in allen modernen Betriebssystemen
  - Fehlen von guten Paket-Scheduling Mechanismen und AQM in den Betriebssystemen, vor allem in kritischen (Edge) Geräten wie Heimnetzwerk-Router, DSLAMs, Kabelmodems, 2G/3G-Basisstationen, Wifi-APs, …
- Viele Messverfahren übersehen Bufferbloat
  - Fällt nicht auf bei konstant hoher Last, nur bei kurzfristigen Ereignissen

Bad Queues und Good Queues

Bad Queue



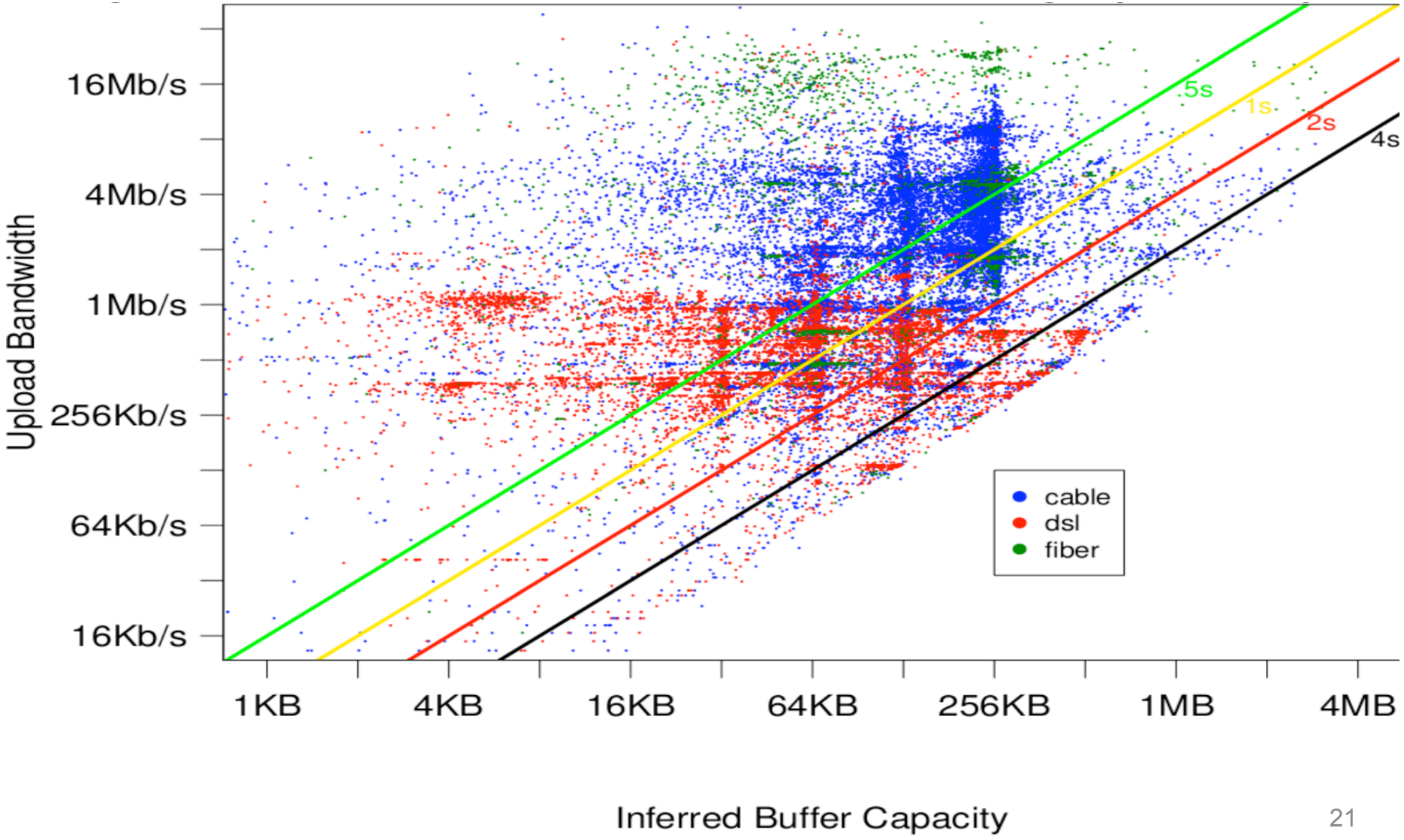
Good Queue
(nur als
„shock absorber“)



Standard Tail Drop Queue Management

- Queues sind nötig um kurzfristig die Leitungsbandbreite übersteigende Übertragungsraten abzufangen
  - (z.B. an Übergängen von hoher zu niedriger Leitungsbandbreite: vor “Bottlenecks”)
  - Zu kleine Puffer resultieren in reduzierten Durchsatz
  - Zu große Puffer zu hoher Paketverzögerung
  - Leitungsbandbreitenunterschiede können heutzutage größer als 5 Größenordnungen sein (~100KBit/s bis 40GigE)
  - Übertragungs-“Bursts“ werden immer größer
- Es gibt keine eine korrekte Puffergröße für alle Fälle
  - Gerätehersteller setzen i.d.R. auf zu große Puffer, statt zu klein
  - Führt zu Latenzen und TCP Fehlverhalten!

Buffer Capacity vs. Bandwidth vs. Latency



Controlled Delay (Codel)

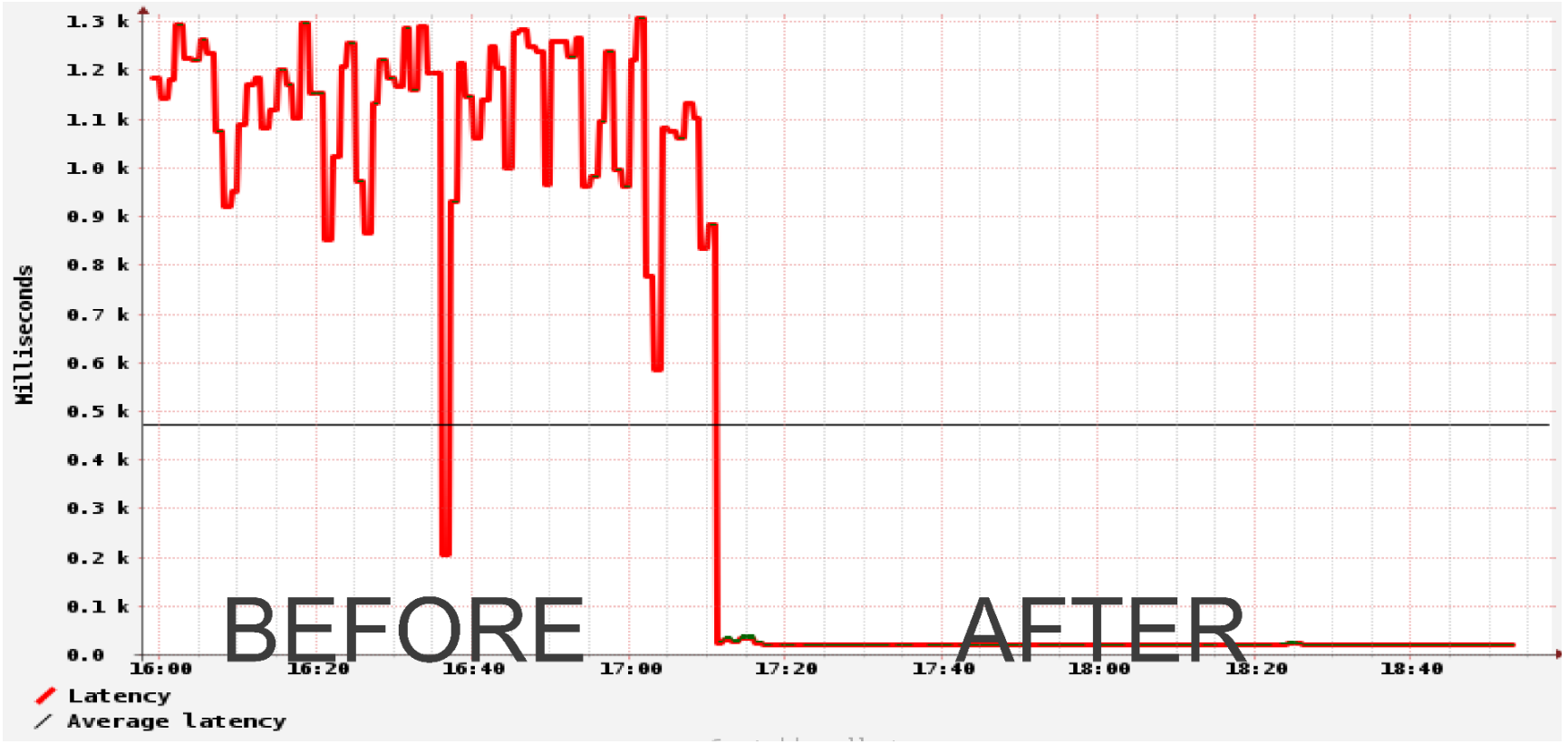
- Misst die Paketverzögerung vom Eingang in die Warteschlange bis zum Austritt
  - Pakete werden am Pufferanfang mit Zeitstempel versehen und beim Verlassen abgelesen
- Maßstab unabhängig von der Leitungsbandbreite, universell einsetzbar ohne Konfiguration
- Wird die Pufferverzögerung (im aktuellen Intervall) zu groß
  - Grenzwert überschritten
  - Verwirf ein möglichst altes Paket (Head Drop, nicht Tail drop!)
  - Falls nicht ausreichend, verwirf Pakete in immer kleineren Intervallen bis Zielverzögerung wieder unterschritten
  - Werte: 100ms initiales Intervall, 5ms Pufferverzögerung
  - Data Center brauchen viel kleinere Werte

fq_codel

In der Praxis (Linux 3.5, IETF Draft): Kombiniere Codel mit SFQ

- Stochastic Fairness Queueing (SFQ)
- Große Zahl von FIFO-Queues, auf die alle Flows aufgeteilt werden
  - z.B. bis zu 64K Queues in Linux 3.4, Standard 127 für DSL
  - Ideal: 1:1 Flow zu Queues, aber hier Aufteilung durch Hash-Funktion auf weniger Queues (daher „stochastic“)
- Round Robin der Queues
- Gleichbehandlung aller Flows
  - Große Flows verdrängen kurze nicht; löst das „Mice and Ant“-Problem
- Codel misst Pufferverzögerung der einzelnen Queues, dropped Pakete in den größten Queues („Elephant Flows“), sonst nicht

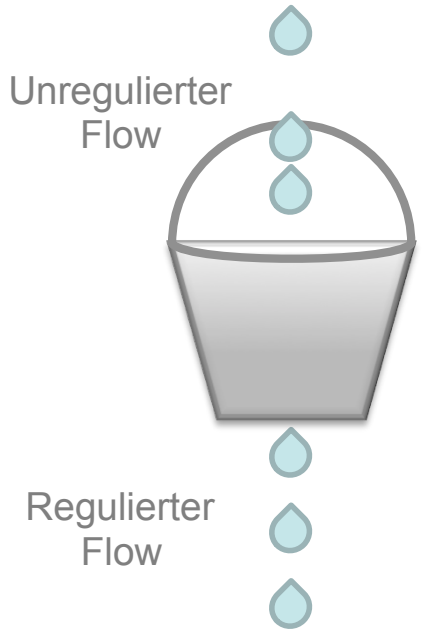
fq_codel


1.3 k
1.2 k
1.1 k
1.0 k
0.9 k
0.8 k
0.7 k
0.6 k
0.5 k
0.4 k
0.3 k
0.2 k
0.1 k
0.0
Milliseconds
BEFORE
AFTER
16:00
16:20
16:40
17:00
17:20
17:40
18:00
18:20
18:40
Latency
Average latency

Shaping

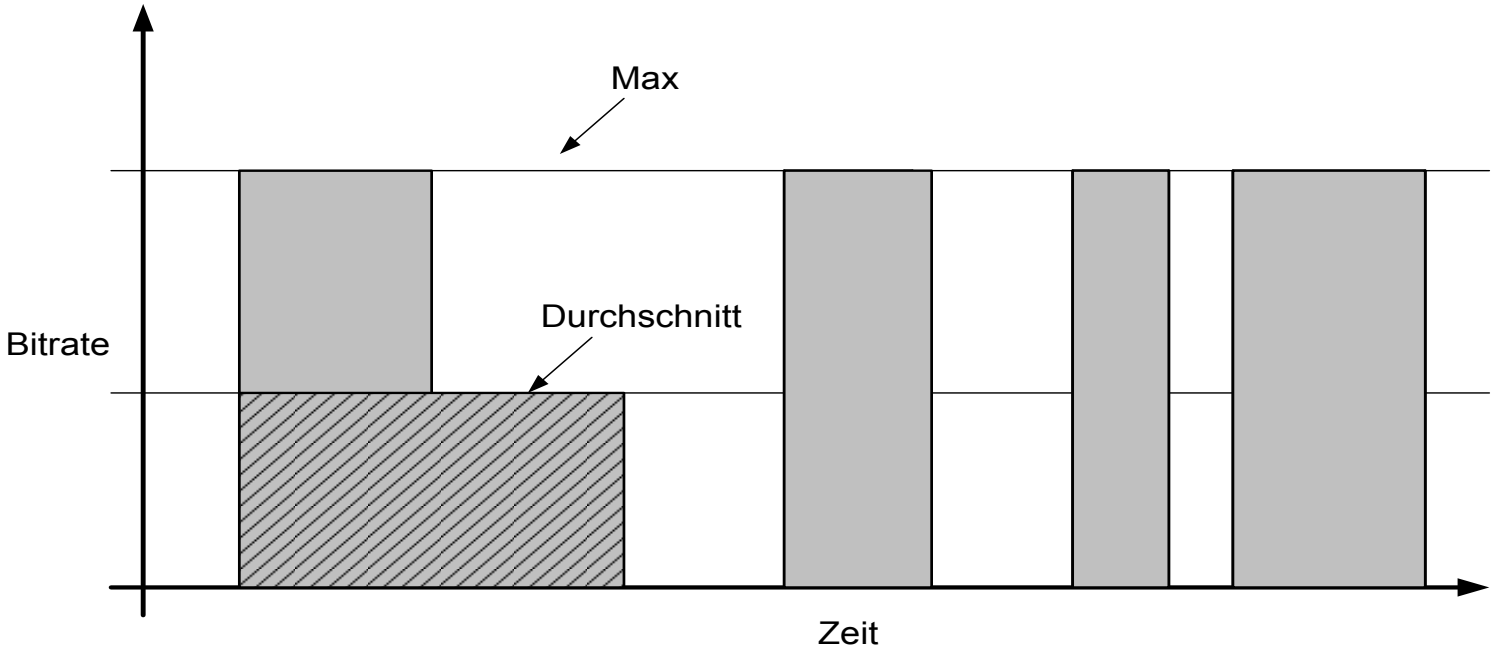
- Ähnlich zu Policer aber mit dem Unterschied, dass die zeitliche Dimension der Paketvermittlung modifiziert wird

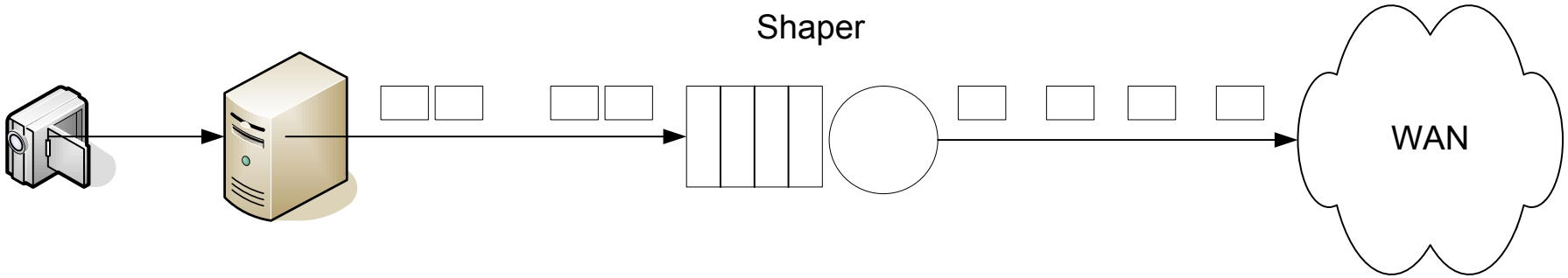
- Beispielshaper: Leaky Bucket
  - „Kübel mit Loch im Boden“
  - **Pakete** fallen in den Kübel (nicht Token!)
  - Tröpfeln mit fixer Rate aus dem Loch



Shaping

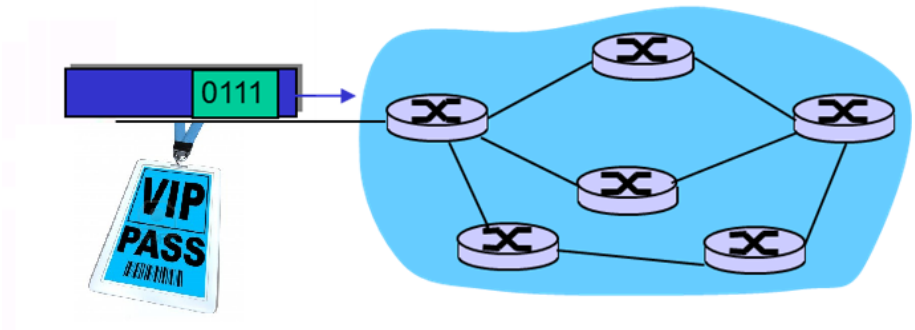
- Soll ein Sender seine Pakete **sofort** abschicken?





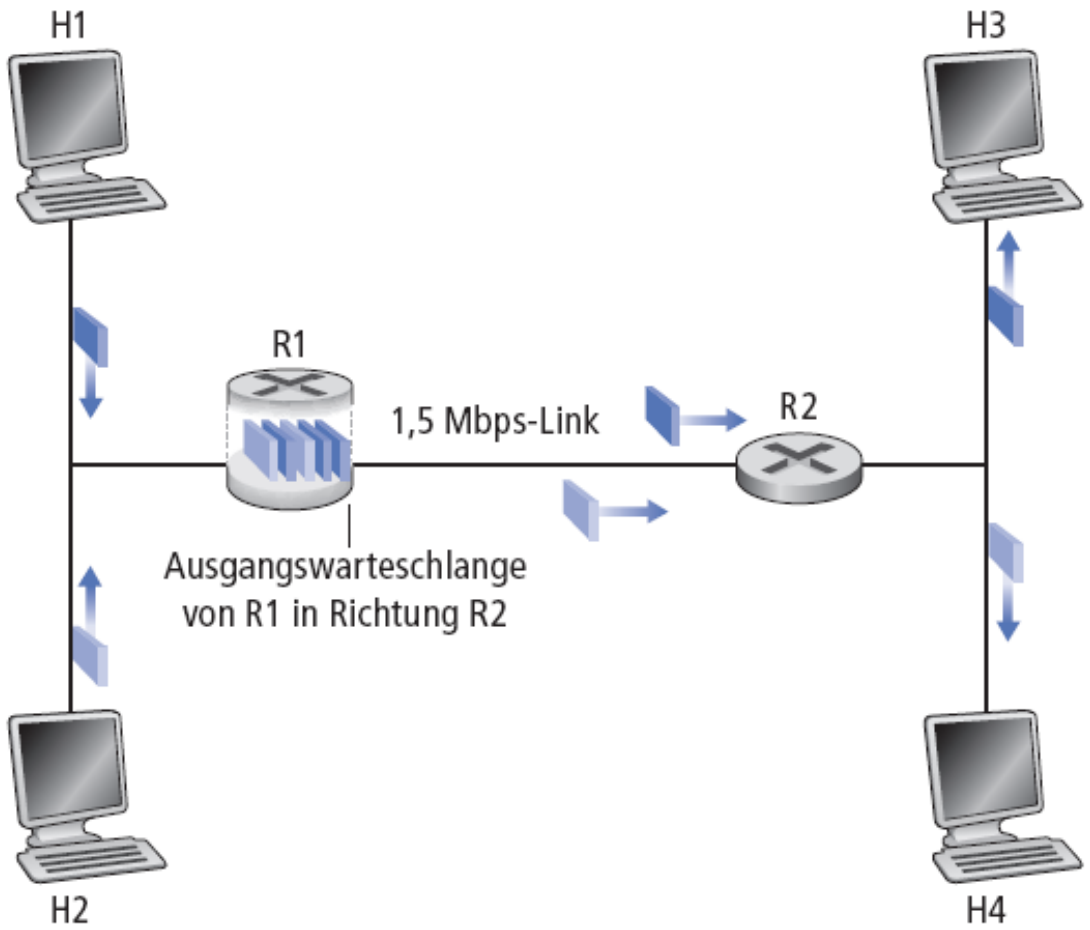
Anbieten von mehreren Dienstklassen

- Best-Effort-Netzwerke: Gleiche Dienstgüte für alle
- Die Alternative: Verschiedene Dienstklassen
  - Aufteilen des Datenverkehrs in Klassen
  - Das Netzwerk behandelt die unterschiedlichen Klassen verschieden (Analogie: VIP-Teilnehmer und normaler Teilnehmer)



Anbieten von mehreren Dienstklassen

Szenario:

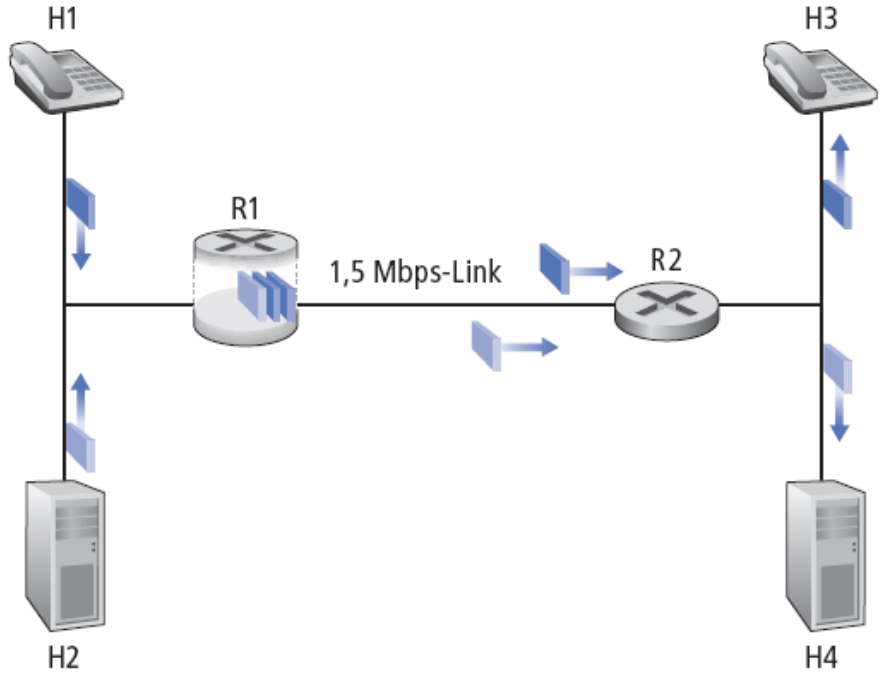


Prinzipien von QoS (1/3)

Beispiel:

1-MBit/s-Videotelefonieverbindung und ein FTP Upload teilen sich einen 1.5-Mbit/s-Link

- FTP kann im Router Überlast erzeugen
- Priorität sollte auf dem Audio liegen

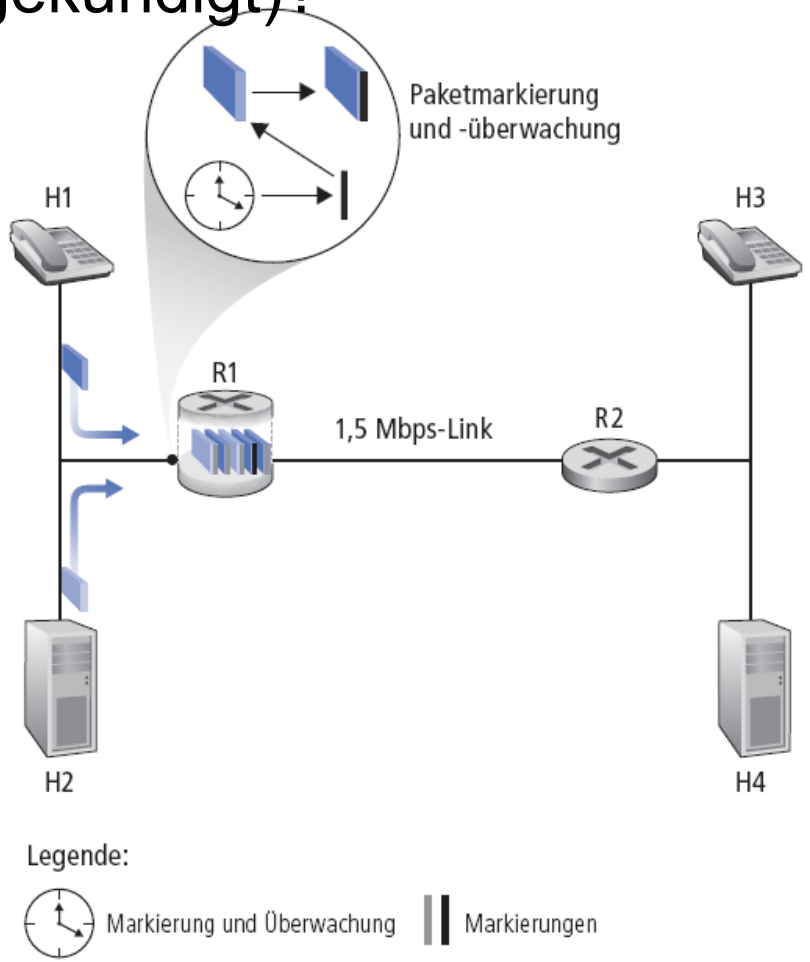


1. Prinzip:
Das Erkennen (Identifizierung) von bestimmten Paketen ist notwendig, damit ein Router zwischen den verschiedenen Verkehrsklassen unterscheiden kann.

Prinzipien von QoS (2/3)

Was passiert, wenn die Anwendungen sich falsch verhalten (z.B. Audio-Rate ist höher als angekündigt)?

→ Policing: Quellen dazu zwingen, sich an die Absprachen zu halten!

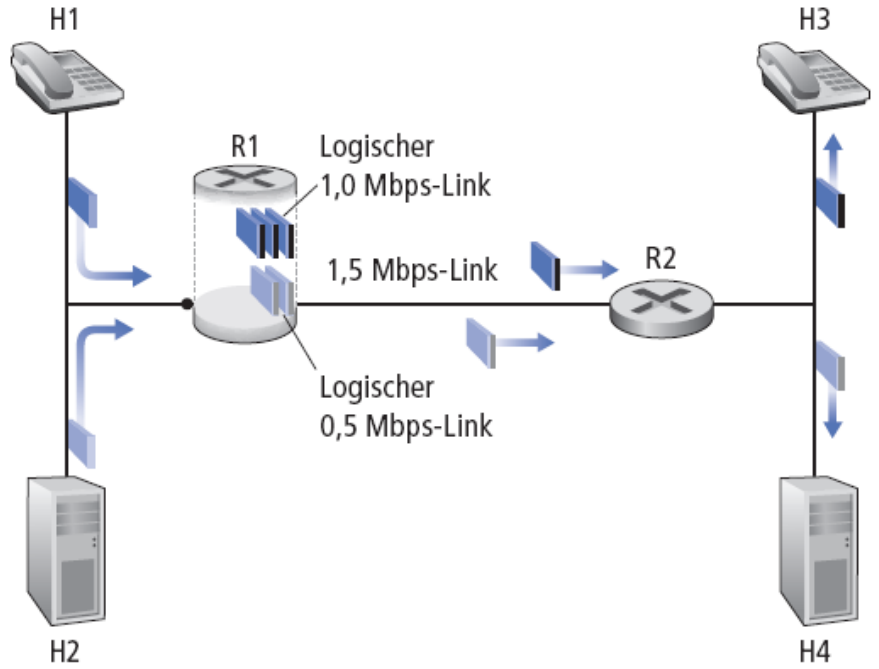


2. Prinzip:
Die logische Trennung (Isolation) der einzelnen Netzwerkströme muss durchgeführt werden, um sie ressourcentechnisch differenziert behandeln zu können.

Prinzipien von QoS (3/3)

Eine feste Zuordnung von Ressourcen zu Netzwerkströmen ist ineffizient, wenn Ströme die eigenen Limits nicht ausnutzen!

3. Prinzip:
Resource Conservation: Trotz Isolation müssen die Ressourcen so gut wie möglich ausgenutzt werden!



Netzwerk-Struktur

Netzwerke haben drei „Facetten“:

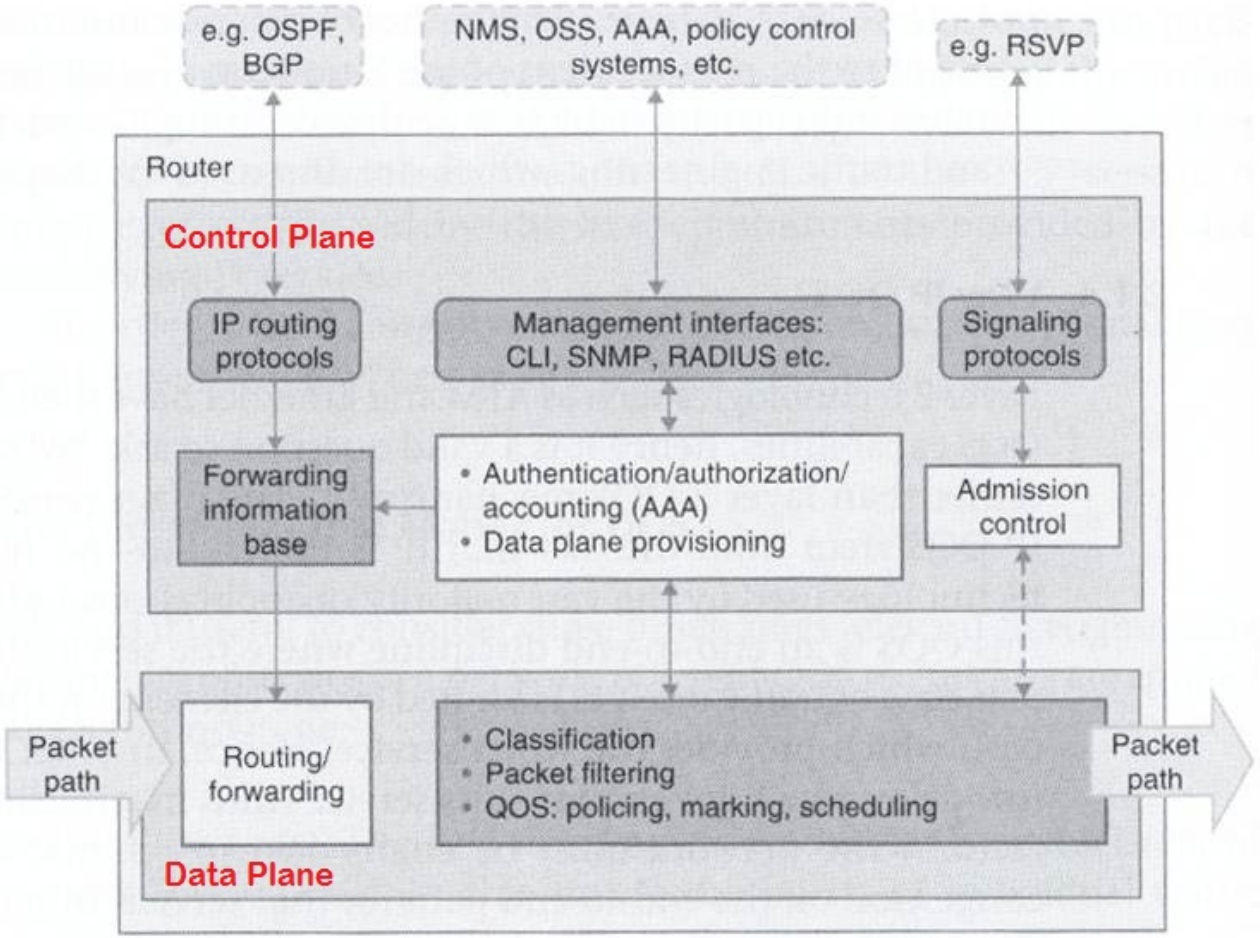
- **Management Plane**
  - Accounting, Billing, Traffic Engineering
- **Control Plane**
  - Wie wird die QoS gesteuert?
- **Data Plane**
  - Wie leitet das Netzwerk reine Datenpakete weiter? Wie werden Pakete differenziert behandelt?
  - Wie wird QoS garantiert?

QoS auf Control und Data Plane

- Data Plane
  - Klassifizierung (Classification)
  - Markieren (Marking)
  - Priorisierung (Prioritization)
  - Maximum Rate Enforcement
  - Minimum Rate Assurance

- Control Plane
  - MPLS (Multiprotocol Label Switching)
  - DiffServ (Differentiated Services)
  - IntServ / RSVP (Integrated Services / Resource reSerVation Protocol)

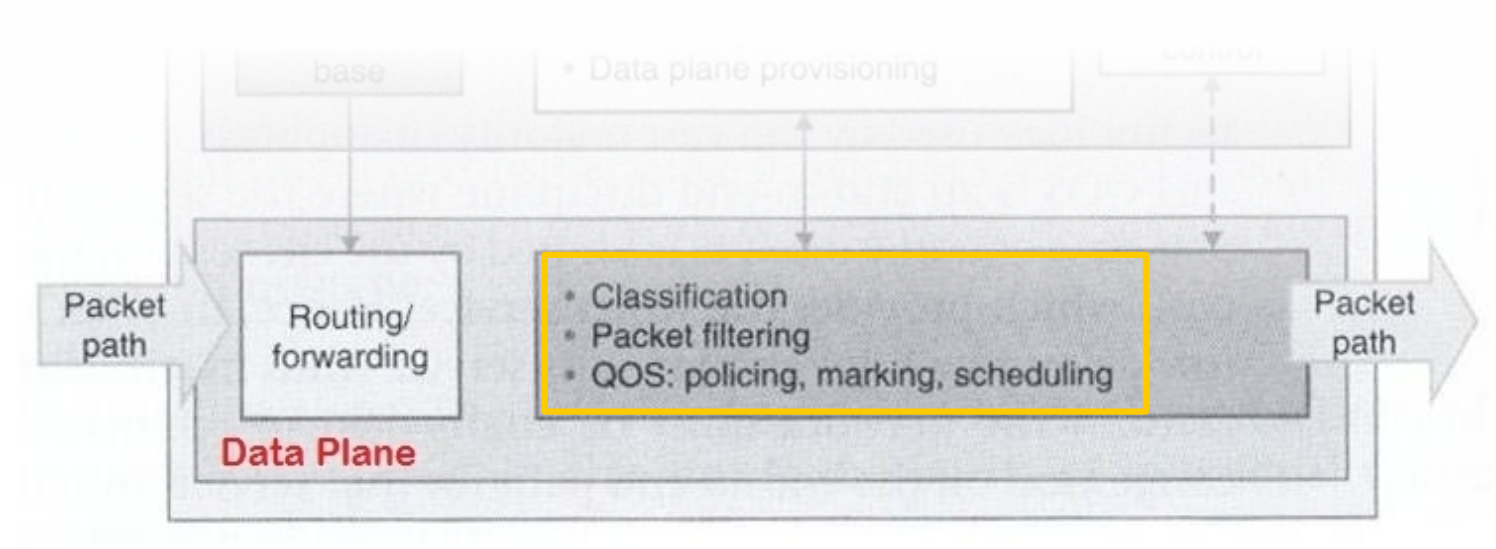
Control und *Data Planes*



4. Quality of Service

- 4.1 Grundlagen
- 4.1 Warteschlangen Management
- **4.2 Data Plane QoS Mechanismen**
- 4.3 Control Plane QoS Mechanismen

Data Plane QoS Mechanismen



Klassifizierung (Classification)

- Microflows
  - z.B. {Source IP, Dest. IP, Source Port, Dest. Port, Protocol}
- Stream: Aggregat von Flows
- Traffic Class: Aggregat von Streams oder Flows

Klassifizierung (Classification)

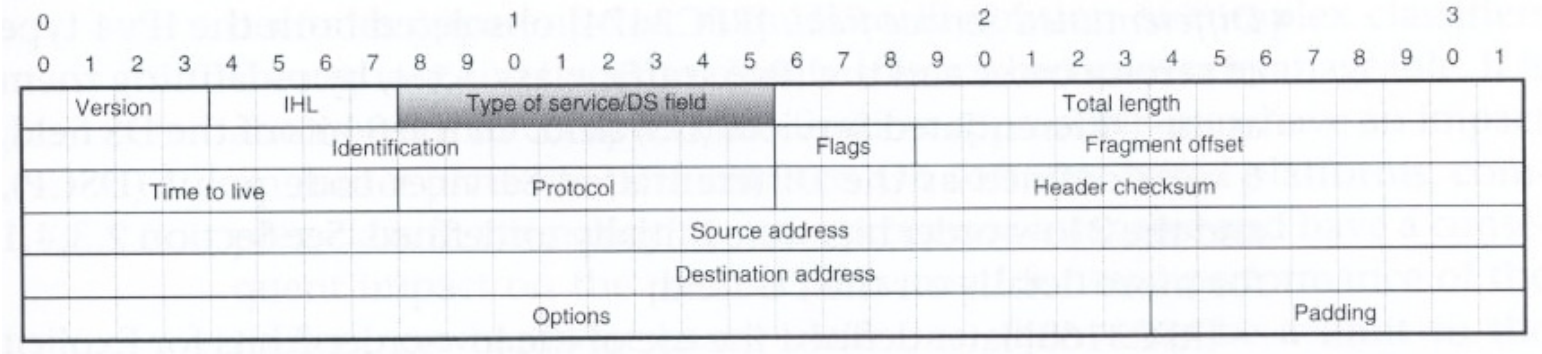
Unterscheidung zwischen den Flows, um diese differenziert behandeln zu können.

Klassifizierungstypen

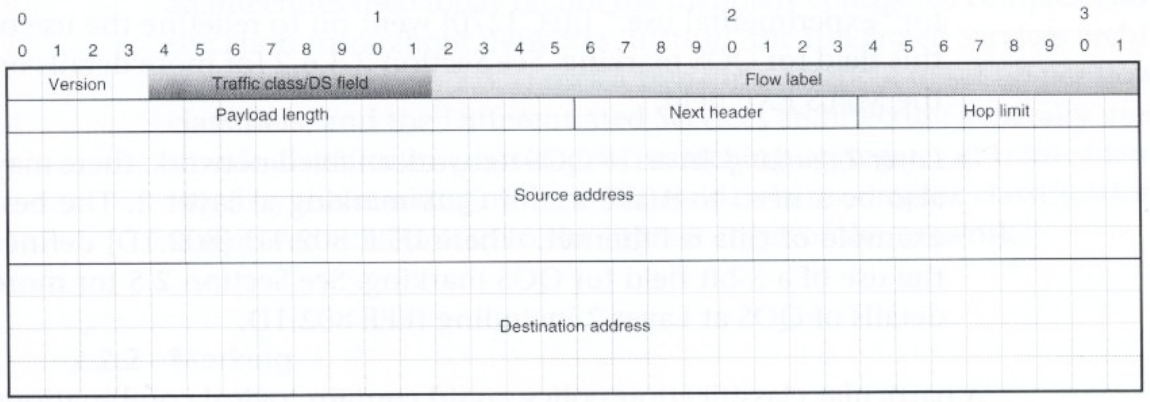
- Implizit
  - Keine Info der Header/Payload erforderlich
    - Z.B.: Von welcher NIC kommt das Paket?
- Einfach
  - IP-Header-Felder sind vorhanden, die explizit für QoS vorgesehen sind
- Komplex
  - Mit Feldern des IP-Headers/MAC-Adresse, die nicht explizit für QoS vorgesehen sind
- Deep Packet Inspection / Stateful Packet Inspection
  - Auswertung der Payload (DPI) oder Auswertung der Verbindungszustandes (SPI)

Simple Classification

- IPv4: Type of Service (ToS) Octet (RFC 791)



- IPv6 Traffic Class Octet (RFC 2460)



→ Später Änderung in **Differentiated Services** (DS) Field (RFC 2474, RFC 3168)

Markieren (Marking)

Wird auch „*Coloring*“ genannt.

→ Setzt Felder für explizite QoS Klassifikation.

- *Source Marking*
  - Sender der Pakete führt das *Marking* durch
    → Kann das Netzwerk allen Sendern vertrauen?
- *Ingress Marking*
  - Geschieht am **Eingang** des Netzwerks, durch vertrauenswürdiges Device
  - Ältere Markings werden überschrieben (= *Re-marking*)
  - Man unterscheidet 2 Arten:
    - **Unconditional**: Hängt von keiner Voraussetzung ab
    - **Conditional**: Hängt davon ab, wie das Paket bewertet wird (z.B. von einem *Policer*)

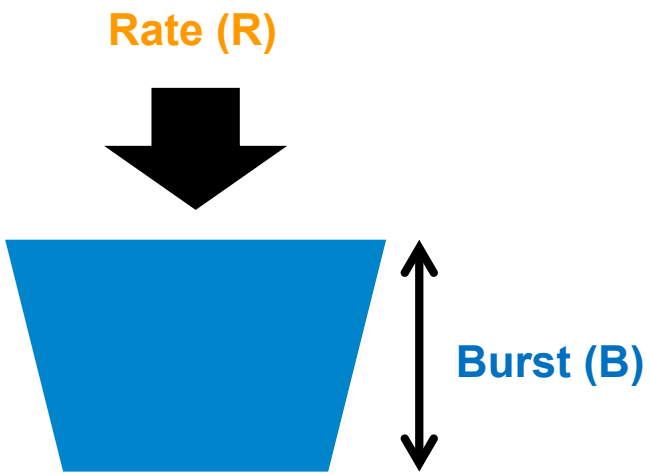
Policing – Token Bucket

Ziel:

- Datenverkehr so einzuschränken, dass er die vereinbarten Parameter nicht überschreitet. Die Pakete werden dementsprechend markiert.

Policer werden in Form von Token Buckets dargestellt

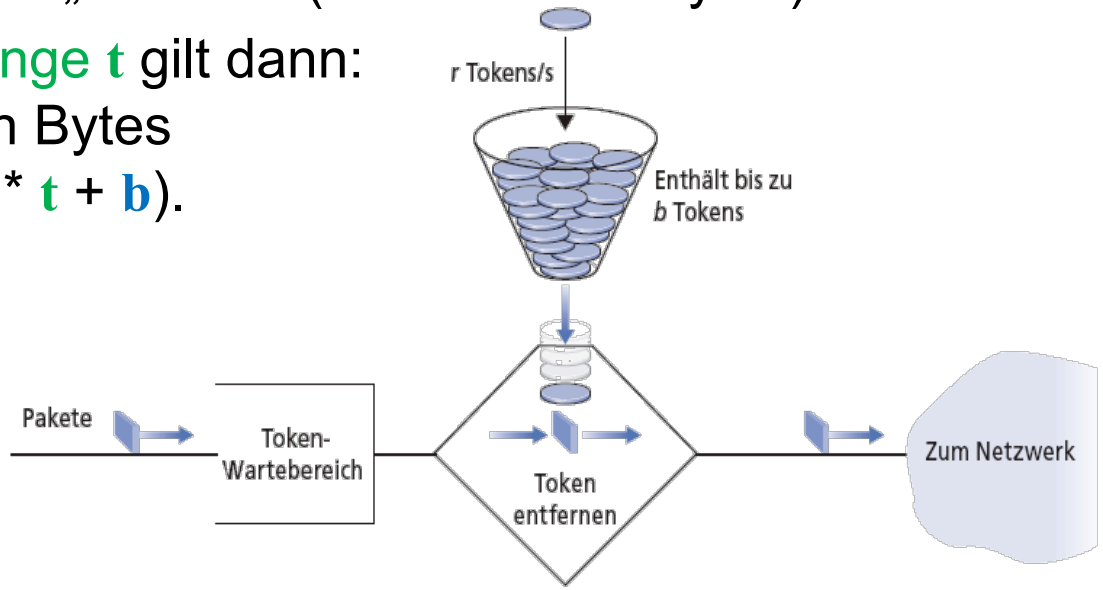
- Bucket (Kübel) hat ein Fassungsvermögen (**Burst B** Tokens)
- Wird mit Tokens angefüllt (**Rate R** Token/s)
- Jedes Token repräsentiert ein Byte
- Pakete werden vom Policer markiert



Policing – Token Bucket

Token Bucket:

- Der Bucket kann maximal **b** Tokens beinhalten
- Tokens werden mit der Rate **r** Token/Sekunde generiert, solange der Eimer nicht voll ist
- Übertragen eines Paketes „kostet“ x (= Anzahl der Bytes) Tokens
- Über ein Intervall der Länge **t** gilt dann: Anzahl der übertragenen Bytes ist kleiner oder gleich (**r** * **t** + **b**).



Policing – Token Bucket

Drei häufig verwendete Kriterien:

- **Durchschnittliche Rate** (long term average rate):

  Wieviele Pakete können über eine längere Zeit hinweg im Durchschnitt pro Zeiteinheit übertragen werden

- **Maximale Rate** (peak rate):

  Zusätzlich zur durchschnittlichen Rate, um zu bestimmen, wie sich die Datenrate über kürzere Zeiträume verhalten darf.

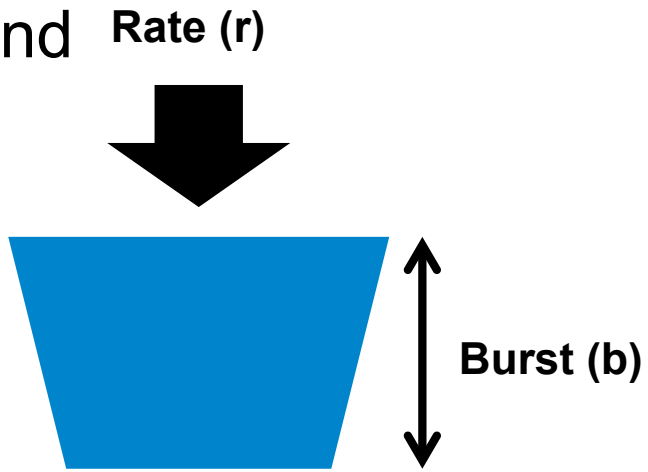
  – *z.B. 6000 Pakete pro Minute durchschnittliche Rate mit einer maximalen Rate von 1500 Paketen pro Sekunde*

- **Burst-Größe**:

  Maximale Anzahl von Paketen, die in Folge (ohne jede Verzögerung) übertragen werden dürfen

Policing – Token Bucket

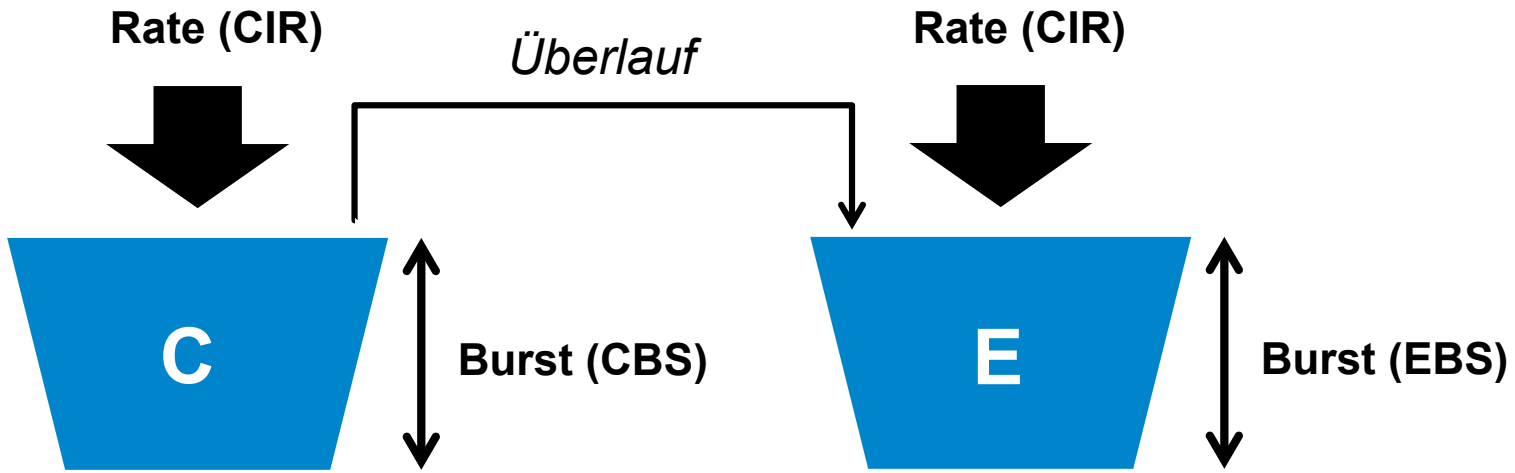
- Wenn genug Tokens im Bucket sind → *Packet conforms*
  - x Tokens werden aus dem Bucket entfernt
  - Paket markieren als „in-contract“
  - Paket wird weitergeleitet
- Wenn nicht genug Tokens im Bucket sind
  → *Packet exceeds*
  - Paket wird verworfen, oder
  - Paket markieren als „out-of-contract“ und weiterleiten (**Marking**)



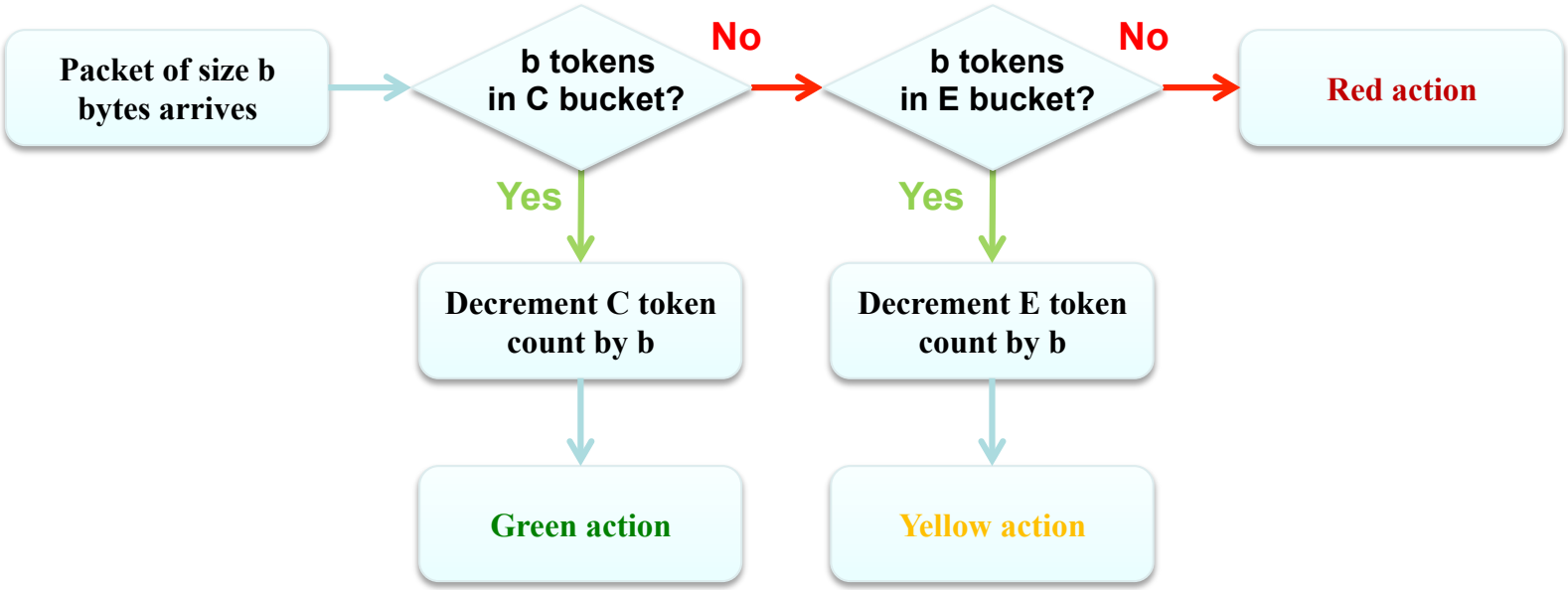
→ **Policer verzögern niemals Pakete** (Shaper schon)**!**

Single Rate Three Color Marker (SR-TCM)

- Definiert in RFC 2697
- Drei Ampel-Farben: Grün, Gelb, Rot
- Zwei Buckets
  - Committed Information Rate (CIR)
  - Committed Burst Size (CBS) und Excess Burst Size (EBS)



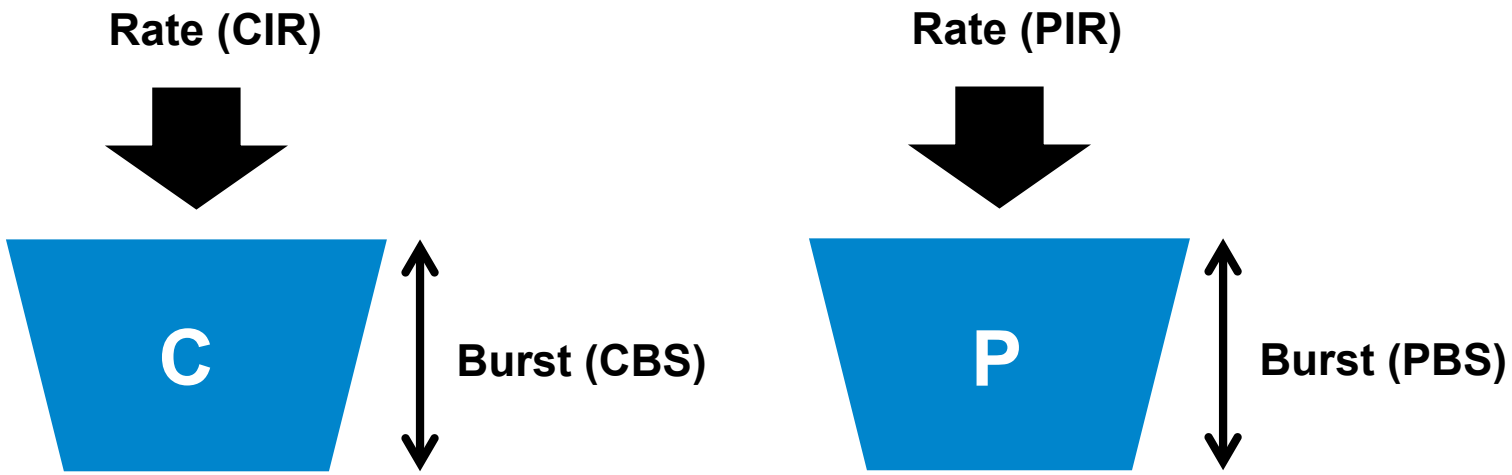
Single Rate Three Color Marker



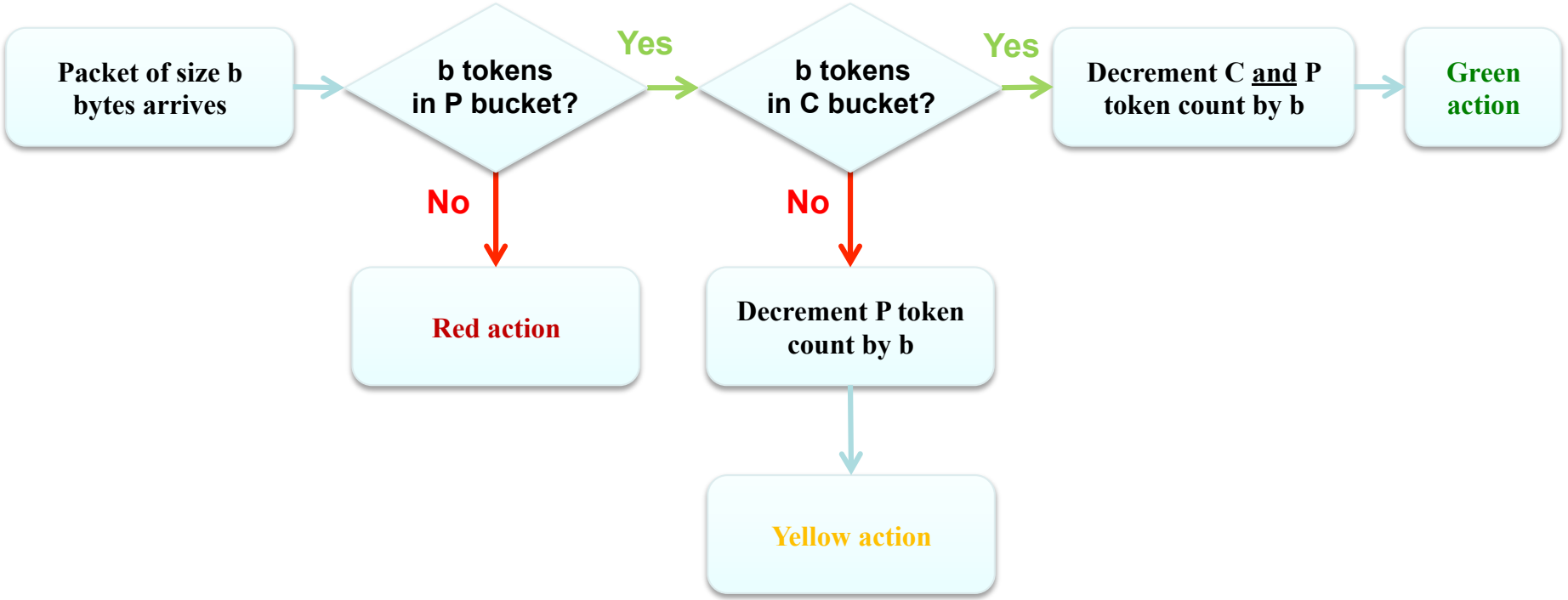
- Green action: Mark in-contract + transmit
- Yellow action: Mark out-of-contract + transmit
- Red action: Drop (or mark exceedingly-out-of-contract + transmit)

Two Rate Three Color Marker (TR-TCM)

- Definiert in RFC 2698
- Zwei Buckets
  - Committed Information Rate (CIR), Committed Burst Size (CBS)
  - Peak Information Rate (PIR), Peak Burst Size (PBS)



Two Rate Three Color Marker



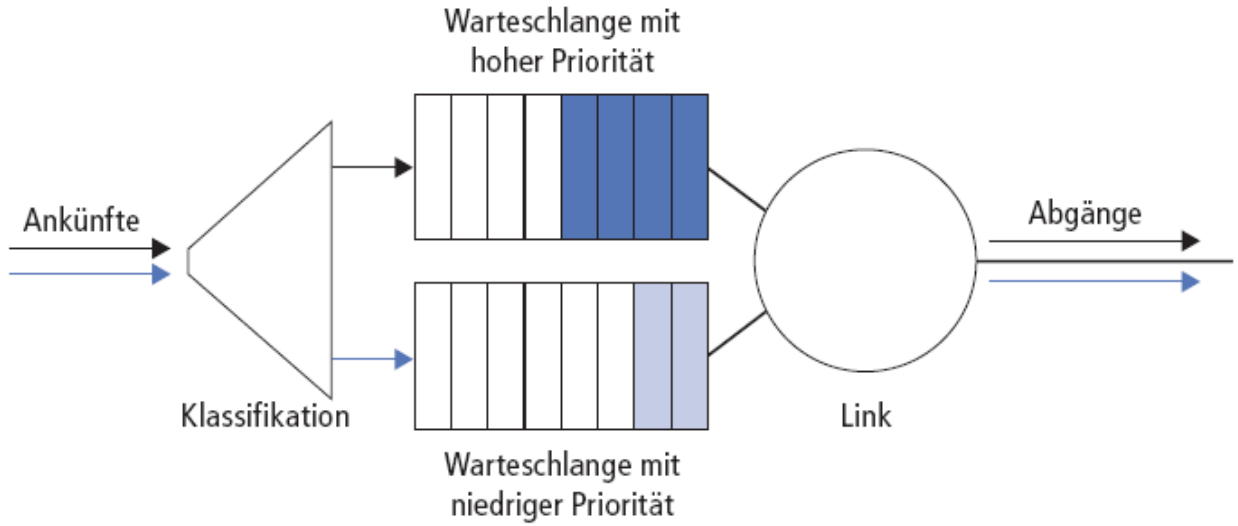
Scheduling und Queuing

- Falls Arbeitslast größer als Ressourcenangebot:

- Scheduler
  - Vermittelt zwischen anfallender Arbeit und Ressourcen
  - Erzeugt einen Schedule
  - ➢ Liste, die angibt, zu welchem Zeitpunkt welches Arbeitspaket verarbeitet werden soll
  - Falls Ankunftsrate größer als Linkrate/Servicerate → Link-Warteschlange füllt sich
  - Regelt welche Pakete den Router wann verlassen sollen

Priority Scheduling

- Prioritätswarteschlange:
  - Übertragen des Paketes mit der höchsten Priorität



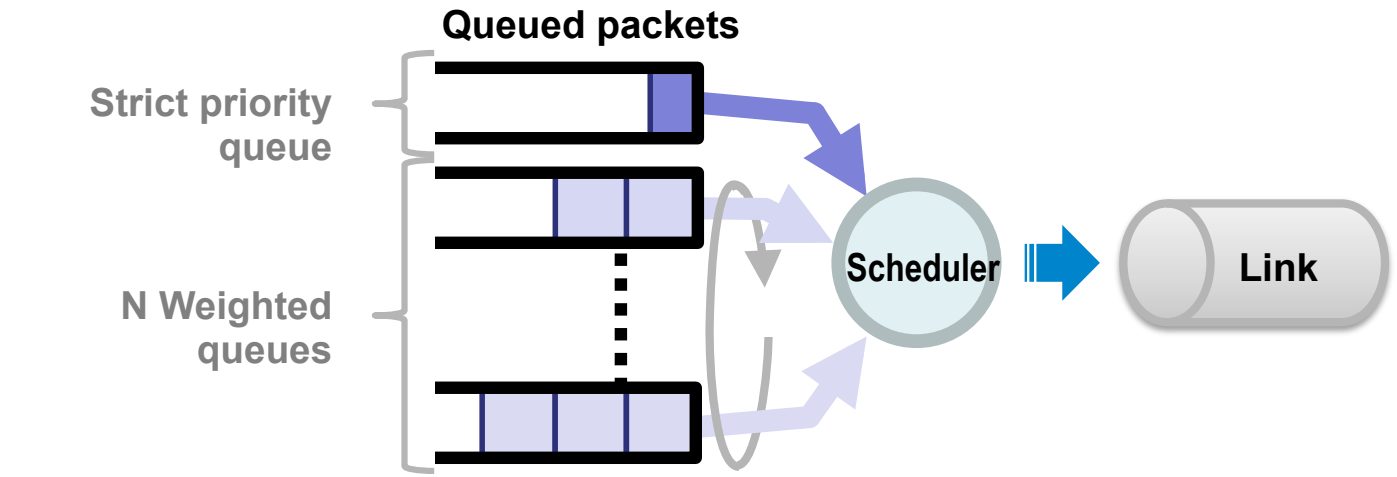
- Mehrere Warteschlangen mit unterschiedlicher Priorität
- Pakete mit höherer Priorität werden immer vorgezogen
- Für Applikationen die nur wenig Delay tolerieren

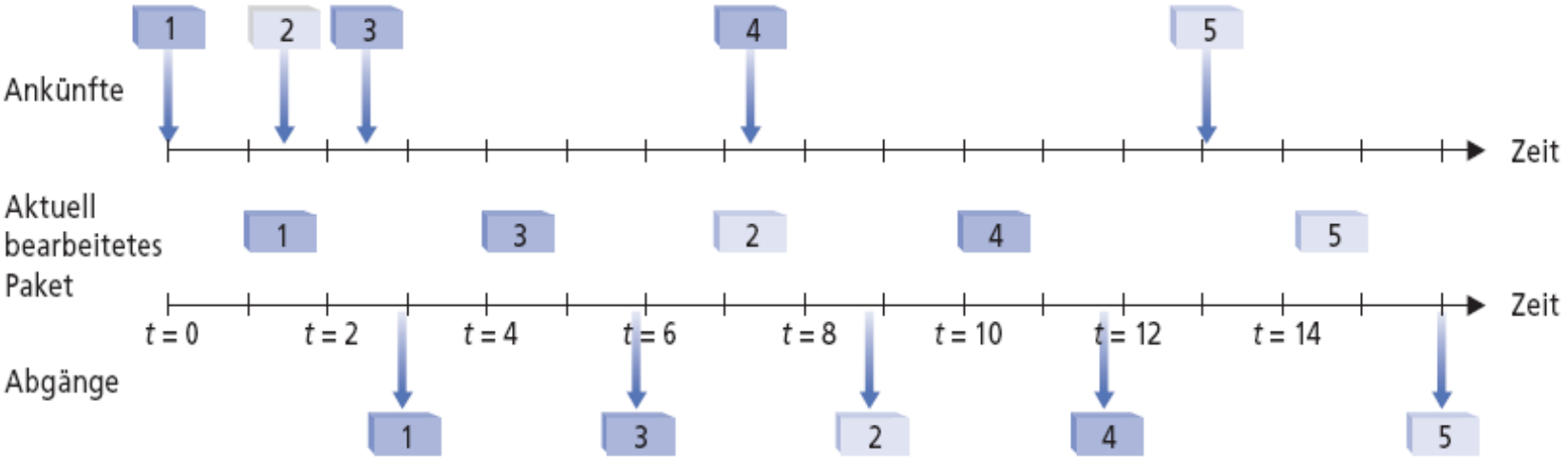
Priority Scheduling

- Pre-emptive
  - Sobald die Priority-Queue aktiv wird, kommt sie dran, unterbricht möglicherweise das Senden eines Pakets aus einer Queue mit niedrigerer Priorität
- Non-pre-emptive
  - Warteschlange mit höchster Priorität kommt als nächste dran, aber keine Unterbrechung der Übertragungen aus den niedrigeren Warteschlangen

- Um eine „Aushungerung“ (engl. Starvation) zu vermeiden wird vor der Priority-Queue gegebenenfalls ein Policer eingesetzt, der z.B. die maximale Datenrate einschränkt

Arbeitsweise einer Prioritätswarteschlange



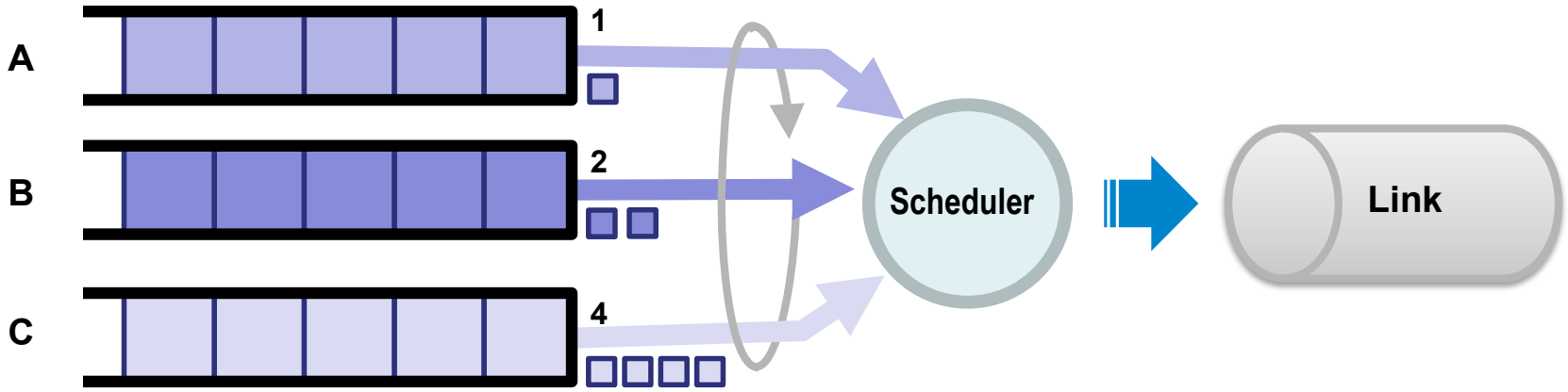


Weighted Bandwidth Scheduling

- Unterschiedliche Warteschlangen
- Keine Priorisierung, sondern Gewichtung
- Gewichte verteilen die verfügbare Bandbreite auf
  - Einzelne Flows oder
  - Verkehrsklassen
- Höheres Gewicht → Warteschlange bekommt mehr Bandbreite

Weighted Round Robin (WRR)

- Scheduler besucht der Reihe nach alle Warteschlangen
- Gewicht = Anzahl der Pakete die vom Scheduler transferiert werden



- Warteschlangen A, B, C, Verhältnis der Gewichte: 1:2:4
- Transferierte Pakete aus Queues: A,B,B,C,C,C,C,A,B,B, …

Fairness

Optimum: Generalized Processor Sharing (GPS)

Soll die Kapazität verstopfter Kommunikationslinks auf effiziente, flexible und faire Weise aufteilen. Dazu wird **Fluid Traffic** (= Pakete werden in unendlich kleine Einheiten geteilt) angenommen. GPS bleibt deshalb leider eine in Netzwerken unerreichbare Ideallösung.

- Fairness
  - Wie nahe kommt die Bandbreitenverteilung dem GPS?
  - Paket-Größen: A ~ 64 Bytes, B ~ 1500 Bytes, C ~ 300 Bytes
  - Bandbreitenaufteilung für das Beispiel aus der vorherigen Folie: **1:47:19**

→ **WRR ist nur fair, wenn alle Pakete gleich groß sind!**

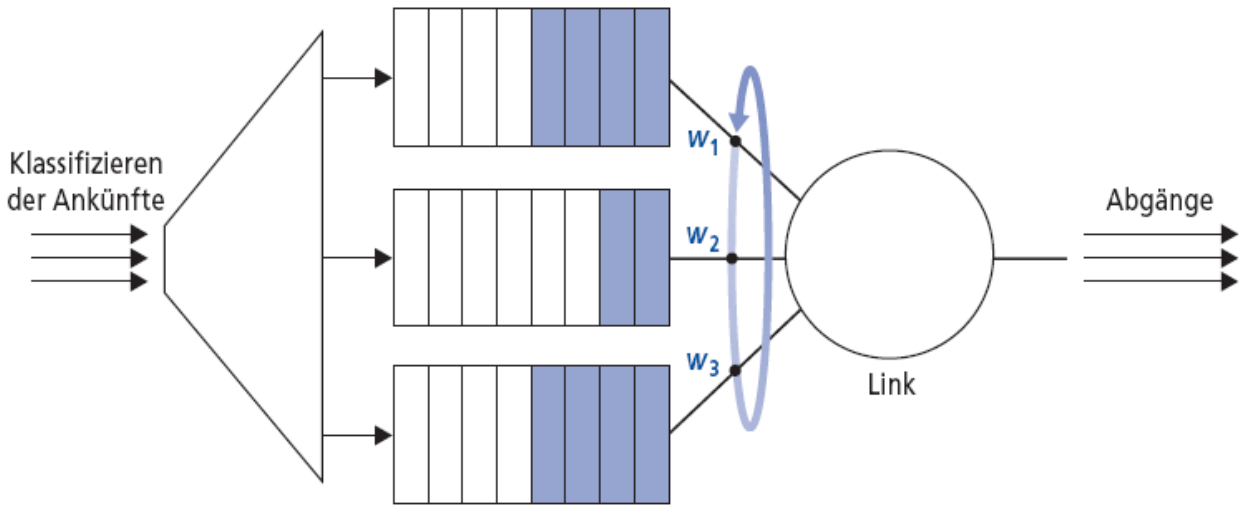
- Falls Pakete ungleich, müssen die Warteschlangen-Gewichte an die durchschnittlichen Paketgrößen angepasst

Deficit Round Robin (DRR)

- Erhöht die Fairness von WRR mit einem Deficit-Counter
- Scheduler besucht die Warteschlangen wie bei WRR
- Weight = Quantum (Bytes) können pro Runde vom Scheduler für diese Warteschlange verarbeitet werden
- Deficit Counter merkt sich für jede Warteschlange, wie viele Bytes sie nicht verwendet hat, diese können bei der nächsten Runde verwendet werden

Weighted Fair Queuing (WFQ)

- Weighted Fair Queuing → neben DRR ein weiterer Ansatz zur Fairness-Verbesserung:
  - Jede Warteschlange hat ein Gewicht
  - Jeden Zyklus wird jede Warteschlange nach ihrem Gewicht bedient



Weighted Fair Queuing (WFQ)

- Pakete bekommen eine Sequenz Nummer
- Sequenz Nummer bestimmt die Paket-Reihenfolge
- Pakete werden dann in dieser Reihenfolge weitergeleitet
- Bsp. Warteschlangen A, B, C, 1:2:4
  → WFQ Gewichte 4, 2, 1

Queues	1. Paket	2. Paket	3. Paket
A	0 + **4***64 = **256**	256 + **4***256 = **512**	
B	0 + **2***1500 = **3000**		
C	0 + **1***300 = **300**	300 + **1***300 = **600**	600 + **1***300 = **900**

- Reihenfolge: A1 (256), C1 (300), A2 (512), C2 (600), C3 (900), B1 (3000), …

4. Quality of Service

- 4.1 Grundlagen
- 4.2 Data Plane QoS Mechanismen
- **4.3 Control Plane QoS Mechanismen**
- 4.4 Transportschicht Mechanismen

Traffic Engineering

Traffic Engineering

ist ein Prozess, der die

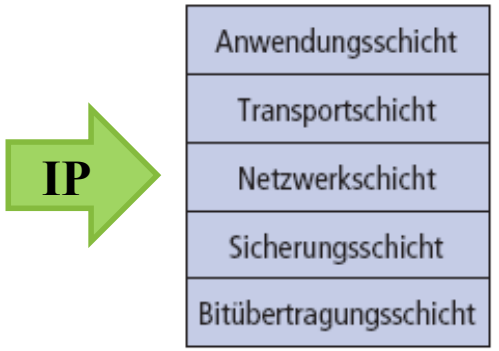
- Erhebung (Analyse),
- Gestaltung und
- Optimierung

von Datenflüssen und -wegen in Netzwerken zum Inhalt hat.

Ziel: Optimales Mapping der Verkehrsanforderungen auf die bestehenden Netz-Ressourcen durch Eingriffe ins Routing.

- Traffic Matrix
  - Wie viele Daten fließen von Ingress *X* zu Egress *Y*?
- Traffic Engineering in IP: **Linkgeweichte** anhand der Verkehrsmatrix und der Topologie bestimmen
- Traffic Engineering mit MPLS: **Pfade** anhand der Verkehrsmatrix und der Topologie anlegen

IP QoS Frameworks – DiffServ



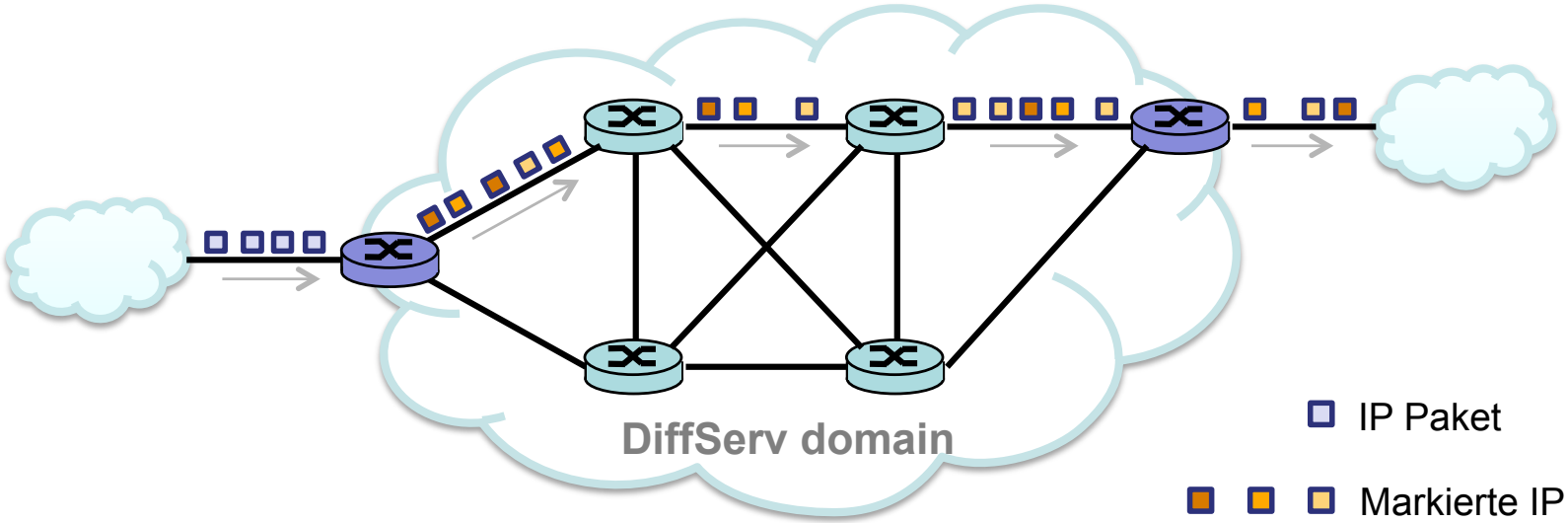
- Differentiated Services (DiffServ)
- Definiert in RFC 2475
- Layer 3 (IP, IP-Router)
- In jeweils einer DiffServ Domain:
  - Traffic wird nach Klassen (Aggregaten) differenziert
  - Aggregate haben QoS-Bedarf: Delay, Jitter, Loss
  - DiffServ skaliert sehr gut, da Router nicht nach einzelnen Flows differenzieren müssen
  - Kein Ende-zu-Ende Einsatz im Internet, sondern wenn überhaupt innerhalb einzelner Netzwerke

Differentiated Services

- Gewünscht sind Dienstklassen
  - Klassen werden unterschiedlich behandelt
  - Klassen können mit unterschiedlicher Priorität behandelt werden: Platin, Gold, Silber

- **Skalierbarkeit:** Einfache Funktionen im Inneren des Netzwerkes, komplexe Funktionen am Rand des Netzwerkes (Hosts oder Zugangsrouter)

Differentiated Services



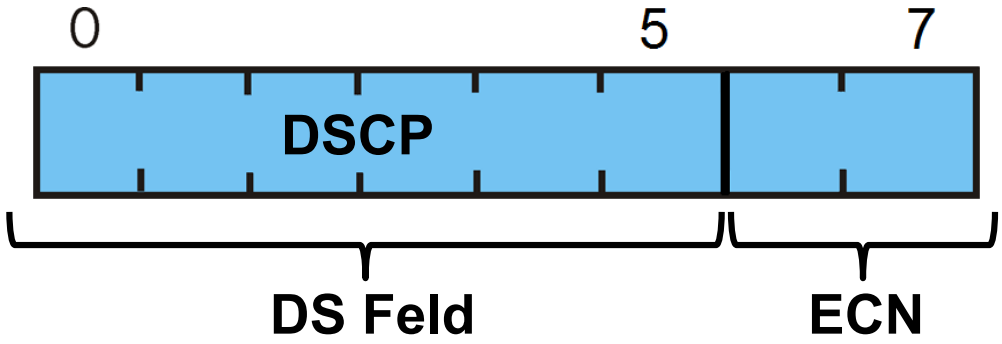
Router am Rand des Netzwerkes:

- Verkehrsüberwachung auf der Ebene einzelner Datenflüsse
- Markiert Pakete danach, ob sie einem gegebenen Profil entsprechen oder nicht

Router im Inneren des Netzwerkes:

- Verkehrsüberwachung auf der Ebene von Verkehrsklassen
- Puffern und Scheduling auf Basis der Markierung
- Bevorzugung von Paketen, die dem Profil entsprechen

Markierung in DiffServ – DS Feld



- Die Markierung erfolgt im Type-of-Service Octet (TOS) bei IPv4 oder im Traffic-Class-Feld bei IPv6
- 6 Bits werden für den Differentiated Service Code Point (DSCP) verwendet
- **DSCP bestimmt welcher Klasse das Paket angehört** und damit das sogenannte Per-Hop Behavior (PHB) im Inneren des Netzwerkes
- Wird am Ingress gesetzt

Per-Hop Behavior (PHB)

- Das Per-Hop-Verhalten ist unterschiedlich für verschiedene Markierungen
- Es führt zu beobachtbaren und messbaren Unterschieden beim Weiterleiten von Paketen
- Es spezifiziert NICHT, welche Mechanismen verwendet werden, um dieses Verhalten zu erreichen

- Beispiele:
  - Klasse A bekommt x% der ausgehenden Linkbandbreite über Zeitintervalle einer fest vorgegebenen Länge
  - Pakete der Klasse A werden immer vor Paketen der Klasse B weitergeleitet

Multi-Protocol Label Switching (MPLS)

- Definiert in RFC 3031
- Verwendet Virtuelle Links – **Label Switched Path (LSP)**:
  Fasst mehrere Layer 2 Links zu einem **virtuellen Pfad** zusammen (ähnlich zu ATM)
- Virtueller Pfad transportiert Verkehrsaggregate
- Geht über *N* Hops
- Nicht End-to-End,
  nur innerhalb eines Autonomen Systems (AS) → Layer 2.5

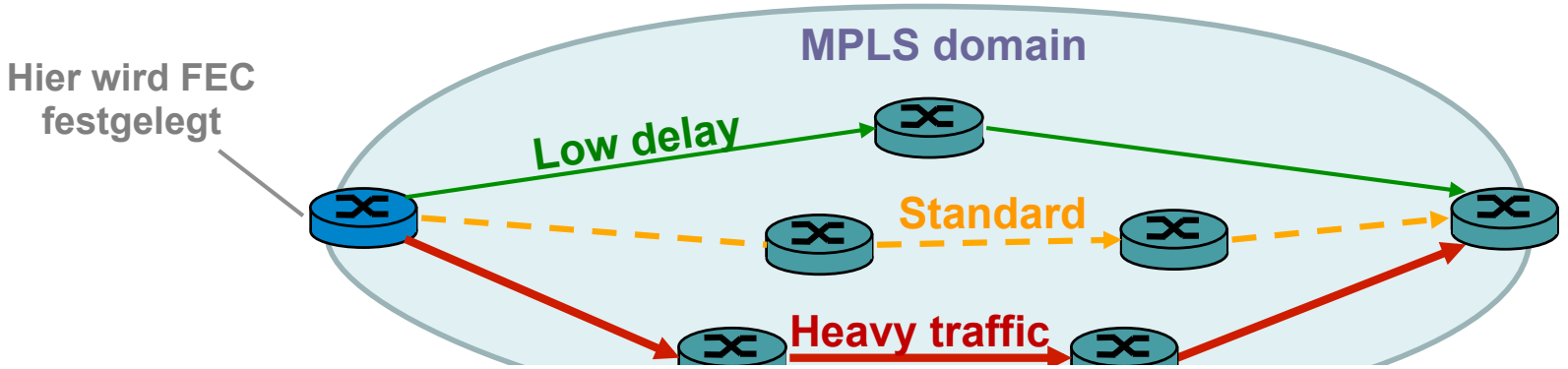
LSPs benötigen keine *IP-Routing Table Lookups*!

Label Switched Path

- MPLS ist **verbindungsorientiert**
- *Per se* keine Bandbreitenreservierung oder harte QoS Garantien
- **Pfade** müssen zunächst **aufgebaut** werden
- Idealerweise Edge-to-Edge (vom Ingress zum Egress eines Netzwerks)
- Router am Eingang (**Ingress**) / Ausgang (**Egress**):
  - **Label Edge Router (LER)**
- Router im Pfad:
  - **Label-Switching Router (LSR)**

Forward Equivalence Class (FEC)

- FEC wird am **Ingress** durch *Label Edge Router* festgelegt, basierend auf:
  - Quell- oder Zieladresse (Netzwerk- oder Hostadresse)
  - Quell- oder Zielport
  - Protokoll-ID
  - Type of Service, DS-Feld
  - IPv6 Flow Label,
  - Usw.
- Unterschiedliche FECs können mittels verschiedener Per-Hop-Behaviors (PHBs) **unterschiedlich behandelt** werden (→ DiffServ)



MPLS und DiffServ

- Definiert in RFC 3270
- Definiert DiffServ über ein darunter liegendes MPLS
- Am Ingress einer MPLS Domain: *Conditioning* und *Labeling* der Pakete
- MPLS untersucht nur das Label, ignoriert aber normalerweise DSCP (im IP-Header)

- Wie wird an den MPLS-Routern differenziert?

MPLS und DiffServ

- EXP-Inferred PHB LSPs (E-LSPs)
  - Ein LSP transportiert mehrere Transportklassen
  - Am Ingress: Abbildung von DSCP (6 Bits, 64 Klassen) auf MPLS Shim-Header EXP-Feld (3 Bits, 8 Klassen)

    → PHB Scheduling Class (PSC)

- Label Inferred PHB
  - Ein LSP transportiert nur genau eine Transportklasse
  - Das Label definiert nicht nur den Pfad, sondern auch die QoS Klasse

Connection Admission Control (CAC)

- ***Admission Control* ist ein Prozess, der in Abhängigkeit von der momentanen Auslastung untersucht, ob ein neuer Flow zugelassen werden kann**
- Wird z.B. bei *DiffServ* am Rande des Netzes durchgeführt, um sicherzustellen, dass die Kapazität im Core-Netz für die verschiedenen Verkehrsklassen ausreicht

Connection Admission Control (CAC)

Wann wird CAC benötigt?

→ Wenn der produzierte Verkehr die verfügbare Kapazität übersteigt

Wann muss/kann/darf man CAC verwenden?

- Die **Service Utility** der Applikationen ist nahe Null, wenn die verfügbare Bandbreite überschritten wird
- Die Art des Services erlaubt es, Requests zurückzuweisen
  (z.B. Funkzelle überlastet)
- Applikationen können auf eine Service-Verweigerung durch die CAC korrekt reagieren

Zusammenfassung: DiffServ

- DiffServ bietet ausschließlich Per-Class-QoS
- Service Level für eine Applikation hängt von den folgenden Fragen ab:
  - Wie viel Kapazität wurde für die Klasse reserviert?
  - Gibt es innerhalb der Klasse *Contention* (d.h. einen Wettstreit)?
- Andere Flows **derselben Klasse** können immer noch die eigene Applikation beeinflussen

  → Keine explizite Reservierung für einen Flow
- Keine zur Gänze sichere QoS-Zusage pro Flow möglich

IETF Integrated Services: IntServ

- Definiert in RFC 1633
- Architektur, um **einzelnen Datenströmen QoS-Garantien anbieten zu können**
- *Service Level Assurance* durch *Bandwidth Management* und *Schedulers*

- **Per Flow** Differenzierung
- **Reservierung von Ressourcen**:
  Router pflegen Zustandsinformationen (Reservierungen)

→ Zentrales Problem, das von IntServ adressiert wird:
Kann ein neuer Datenstrom zugelassen werden, ohne dass die Garantien für andere Datenströme verletzt werden?

IntServ

- *Classification*
  - **Auf jedem** IntServ Router zwischen Source und Destination
  - **Complex Per Flow Classification → *Microflows***:
    {Source-IP, Dest-IP, Source-Port, Dest-Port, Protocol}
  - *Per Flow* Speicherung des Status

- *Scheduling*
  - ***Per Flow* Resource Management** auf jedem IntServ Router zwischen Source und Destination
  - **Queue** entweder für jeden einzelnen Flow oder für **Flow-Klassen**

- *Admission Control*
  - CAC auf jedem IntServ Router für jeden Flow:
    Neuer Flow wird nur dann zugelassen, wenn genug Ressourcen auf dem gesamten Pfad übrig sind

Resource Reservation Protocol (RSVP)

- Definiert in RFC 2205
- Protokoll zur **Reservierung** von **Netzwerk Ressourcen für IntServ**

- Verwendet weder UDP noch TCP
- Hat eigene IP-Protokoll-Nummer (46)
- Für **unidirektionale** Reservierungen (simplex)
- **Unterstützt Unicast** und **Multicast**
- IPv4 und IPv6

RSVP

Was RSVP **NICHT** macht:

- Festlegen, ***wie*** Ressourcen reserviert werden
  → Sondern: ein Mechanismus, um die Anforderungen dem Netzwerk mitzuteilen

- Festlegen, ***welche Route*** die Pakete nehmen sollen
  → Das ist Aufgabe der Routingprotokolle
  → Signalisierung ≠ Routing

- Direkt in das Weiterleiten (Forwarding) von Paketen eingreifen
  → Trennung der Kontrollebene (Signalisierung) von der Datenebene (Forwarding)

RSVP – Eigenschaften

- Kein eigenes RSVP-Routing
  - Verlässt sich auf die verfügbaren Routing-Protokolle

- **Soft State**
  - **Reservierungen (*Path-* / *Resv*-Messages) müssen regelmäßig bestätigt werden, sonst werden sie gelöscht (Timeout)**
  - Dynamische Adaptation für Änderungen bei
    - Multicast Gruppenmitgliedern
    - Netzwerktopologie
    - Änderung einfach durch Schicken von neuen *Path-* / *Resv*-Messages

Path Discovery

- Vor der Reservierung wird eine Path-Discovery von **Sender** zu **Empfänger** durchgeführt (*Path*-Message)
- Pfad von Sender zu Empfänger wird in den RSVP-Routern gespeichert (es gibt aber noch **keine** CAC zu diesem Zeitpunkt!)
- RSVP Router merken sich für **jeden Sender** den RSVP-Router, von dem die Path-Message gekommen ist (Next Hop Richtung Sender [*upstream*])
- **Reservierung** folgt dann dem **Pfad zurück** vom **Empfänger** Richtung **Sender**
- Bei Multicast: Endet erfolgreich, wenn der Multicast-Tree gefunden wurde und dort für diese Session schon mindestens genauso viele Ressourcen reserviert wurden

Vergleich *DiffServ* / *IntServ*

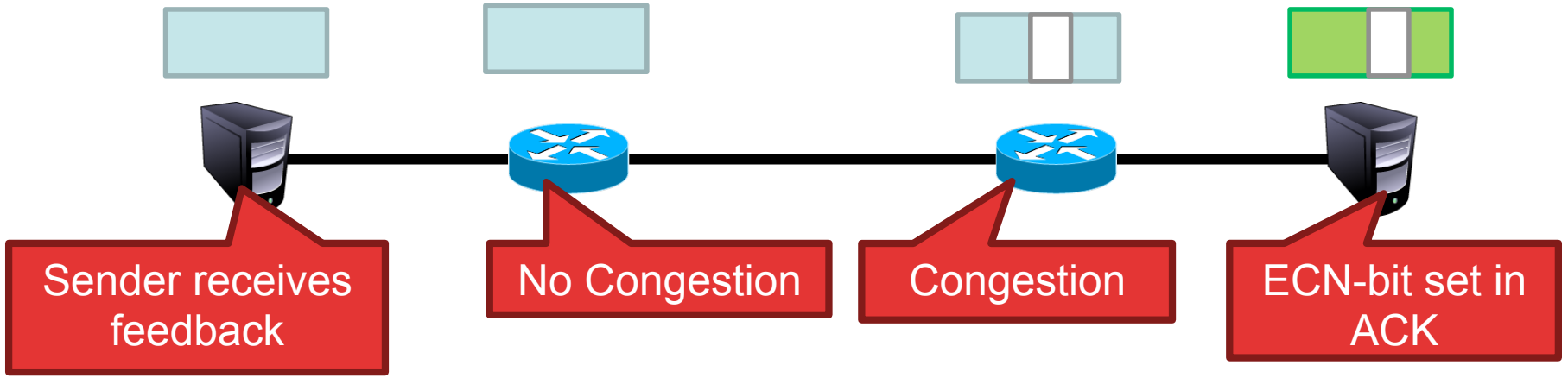
- **Skalierbarkeit und Granularität**
  - *DiffServ* behandelt Verkehrsaggregate und bedarf keiner Flow-Level-Zustandsinformationen in den Netzelementen → gute Skalierbarkeit
  - *IntServ* + RSVP behandeln die Verkehrsanforderungen im Netz einzeln und gehen mit einer Speicherung von Flow-Level-Zustandsinformationen im Netz einher → schlechte Skalierbarkeit bei vielen Flows
- **Einsatzgebiete von *IntServ* und *DiffServ***
  - *DiffServ* ist sehr gut für Kernnetze (Core Networks) geeignet, da es keiner Flow-Level-Zustandsinformationen bedarf
  - *IntServ* is gut für Zugangsnetze (Access Networks) geeignet, da es eine sehr präzise QoS-Differenzierung der einzelnen Verkehrsflüsse ermöglicht
    - Aufgrund einer geringeren Anzahl von Flows stellt die Skalierbarkeitsschwäche von *IntServ* in Zugangsnetzen kein so großes Problem dar wie in Kernnetzen
    - *IntServ* im Zugangsnetz kann auch als CAC-Mechanismus für ein *DiffServ*-Kernnetz eingesetzt werden

4. Quality of Service

- 4.1 Grundlagen
- 4.2 Data Plane QoS Mechanismen
- 4.3 Control Plane QoS Mechanismen
- **4.4 Transportschicht Mechanismen**

Explicit Congestion Notification

- ECN ist ein AQM Mechanismus
- Benutzt TCP/IP Header um ECN Benachrichtigung zu senden
  - Router setzt ECN Flag im Header wenn Überlast auftritt
  - Sendert behandelt ECN markierte Pakete genauso wie verlorene Pakete (Reduzierung des Überlastfensters)
    - Aber es werden keine Pakete verloren



ECN Implementation

- Benutzt 1-bit Feld im TCP Header
  - Congestion Window Reduce (CWR): Reduziere Überlastfenster
  - ECN-Echo (ECE): Feedback im ACK
- Benutzt 2 bits im IP Header ToS Feld
  - 01/10 – ECT, signalisiert ECN Fähigkeit
  - 11 – CE, Signalisiert Überlast
- Unterstützt durch…
  - Windows (Vista+), OS X (10.5+), Linux
  - Cisco routers, *BSD
  - … aber normalerweise deaktiviert

Ist das Internet ECN-fähig?

- Leider nicht wirklich, weil häufig deaktiviert:
  - „Measuring the State of ECN Readiness“ (IMC 2011)
    - ECN incompatible servers: 83-86%
    - ECN incompatible clients: 96-100%
    - … but results are better than for 2004 and 2008
- Viele Router löschen / verwerfen ECN bits
  - Besonders ISP Border Router
  - Alte Router leeren IP ToS Feld und verhindern somit ECN

DCCP

- Datagram Congestion Control Protocol
  - Spezifiziert in RFC4340
  - Eigenständiges Transportschicht Protokoll
  - Unterstützte Funktionalität
    - Für Applikationen mit zeitkritischer Übertragung
    - Ein Datenstrom, nachrichtenorientiert
    - Nutzung von Bestätigungen für:
      - Information ob Paket angekommen ist
      - Signalisierung von Überlast durch ECN
  - Implementiert im Linux Kernel ab 2.6.14, sowie als User-space Implementierung

SCTP

- Stream Control Transmission Protocol
  - Kombiniert Funktionalitäten von UDP und TCP
  - Definiert in RFC4960, implementiert in vielen gängigen OS
  - Funktionalität
    - Multihoming
      - Kommunikation zwischen Endpunkten mit mehr als einer IP Adresse
      - Transparentes wechseln der Netzwerkpfade im Fehlerfall
      - Dynamisch hinzufügbare, entfernbare Adressen währen der Verbindung
      - Dynamische Auswahl des Primärpfades
    - Mehrere Parallele Streams
    - Empfänger kann wählen ob Nachrichten in Reihenfolge des Sendens oder des Empfangens bearbeitet werden

QUIC

- Quick UDP Internet Connections
  - Experimentelles Transportschicht Protokoll
    Entwickelt von Google (2013)
  - Funktionen:
    - Mehrere gleichzeitige Datenströme
    - Verschlüsselungsverfahren ähnlich wie TLS/SSL
    - Reduzierung des Verbindungs-Overhead auf (Ziel: 0-RTT)
  - Ziel: Integrierung in TCP und TLS

QUIC - Funktionsweise

- Reduzierung der Aufbauzeit von Verbindungen
  - QUIC Clients senden die Verbindungsparameter mit dem ersten Datenpaket
  - QUIC Server veröffentlichen eine statische Konfigurationsdatei
  - Nach der ersten erfolgreichen Verbindung speichert der Client ein „synchronization cookie“ des Servers
  - Nachfolgende Verbindungen erfordern dann 0-RTT zum Aufbau
- Reduzierung der Fehlerkorrekturzeit
  - Verwendet auch FEC zur Fehlerkorrektur
  - Proaktive moderne Verfahren anstatt von Überlastfenstern
    - „packet pacing“ – Gleichmäßige Paketübertragung um Überlast zu vermeiden
    - Proaktive Retransmissions von Paketen mit wichtiger Information (Fehlerkorrektur, Verschlüsselungsaustausch)

Übersicht Transportschicht Protokolle

Funktion	UDP	TCP	Multipath TCP	SCTP	DCCP	QUIC
Verbindungsorientiert	Nein	Ja	Ja	Ja	Ja	Ja
Verlässliche Übertragung	Nein	Ja	Ja	Ja	Nein	Ja
Unverlässliche Übertragung	Ja	Nein	Nein	Ja	Ja	Nein
Reihenfolgeerhaltend	Nein	Ja	Ja	Ja	Nein	Ja
Nicht-Reihenfolgeerhaltend	Ja	Nein	Nein	Ja	Ja	Nein
Flusskontrolle	Nein	Ja	Ja	Ja	Nein	Ja
Überlastkontrolle	Nein	Ja	Ja	Ja	Ja	Ja
ECN	Nein	Ja	Ja	Ja	Ja	Nein
Multi-Homing	Nein	Nein	Ja	Ja	Nein	Nein
Multiple-Streams	Nein	Nein	Ja	Ja	Nein	Ja