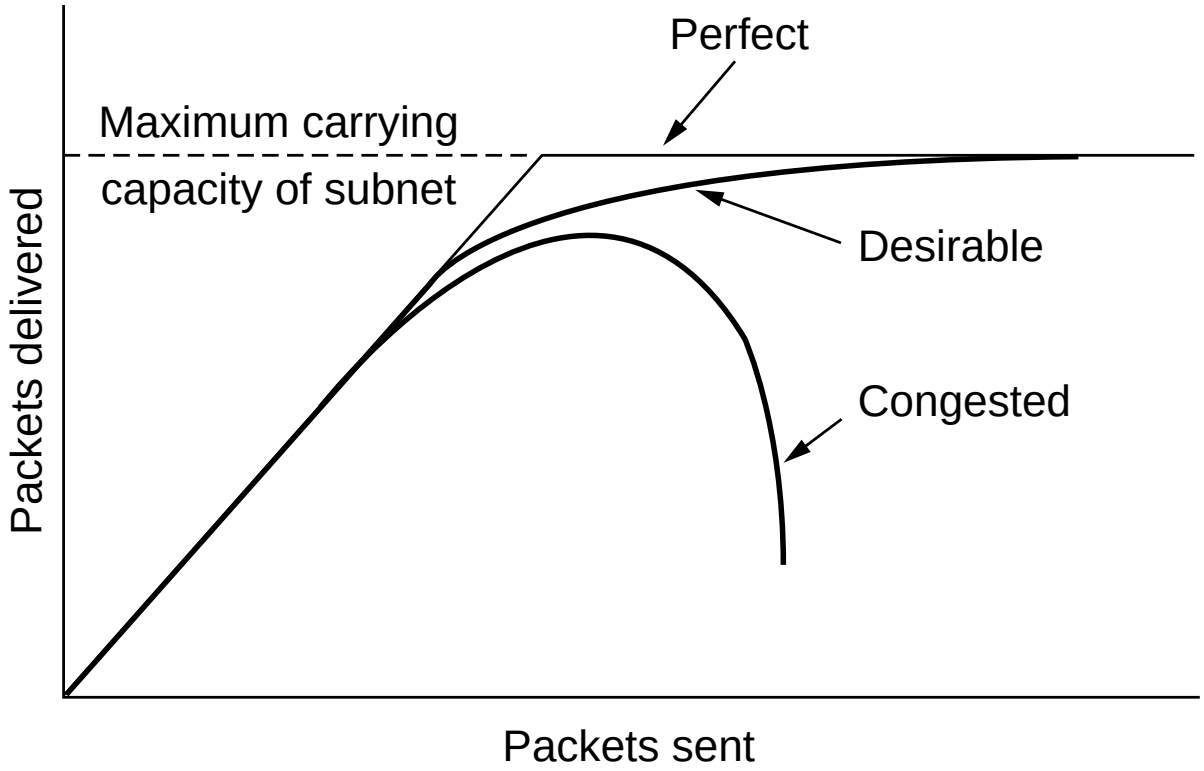
# Congestion Control

# Einleitung

Wenn zu viele Pakete in einem (Teil-)Netz vorhanden sind, kommt es zur Überlastung (Congestion).




(c) Tanenbaum, Computer Networks

# Einleitung

Wenn deutlich weniger Pakete als die maximale Kapazität vorhanden sind,

- erreichen alle Pakete den Empfänger
- Paketverluste nur durch Übertragungsfehler
- lineare Abhängigkeit zwischen gesendeten und empfangenen Paketen

Wenn die maximale Kapazität (fast) erreicht oder überschritten wird,

- werden die Router überlastet und können Pakete nicht mehr weiterleiten
- Paketverluste im Router
- kann bis zum Kollaps des Netzes führen

# Ursachen

- zu geringe Bandbreite
- zu geringe Rechenleistung im Router
  - Prozessor im Router kann seine Kontrollaufgaben nicht schnell genug abarbeiten
  - es bildet sich eine Warteschlange trotz ausreichender Bandbreite
- zu geringer Speicher im Router
  - zu viele gleichzeitig am Router ankommende Pakete können nicht gespeichert werden
  - zu viel Speicher kann auch Nachteile haben, da die Warteschlangen zu lang werden können. Dieses kann Time-outs und Duplikate erzeugen

Es ist eine gemeinsame Optimierung notwendig. Die Optimierung einer einzelnen Komponente verschiebt in der Regel nur den Flaschenhals.

# Unterschied Congestion Control und Flow Control

Congestion Control

- berücksichtigt das gesamte (Teil-)Netz inklusive der einzelnen Komponenten (Hosts, Router,...) und deren Prozesse
- kontrolliert den gesamten Datendurchsatz

Flow Control

- betrachtet point-to-point Verbindungen
- vermeidet Überlastung langsamer Empfänger durch schnelle Sender

Beide können einen “slow down” Request für Sender erzeugen.

# Open Loop Congestion Control

Open Loop Congestion Control

- Vermeidung von Congestion durch richtiges System Design
- Auswahl der richtigen Komponenten, Regeln, Prozesse, ... vor der Inbetriebnahme
- berücksichtigt nicht den aktuellen Zustand des Netzes, keine Messung oder Steuerung
- Unterscheidung zwischen Open Loop Congestion Control am Sender und am Empfänger

# Closed Loop Congestion Control

Closed Loop Congestion Control

- reagiert auf den aktuellen Zustand des Netzes mit einem Feedback Kanal
- Detektion wann und wo die Congestion auftritt
- Verbreiten dieser Information zu den steuernden Komponenten
- Anpassung des Systems um das Problem zu beheben
- explicit feedback: Dedizierte Pakete mit Warnungen
- implicit feedback: Detektion durch lokale Messungen

# Mechanismen

Messung und Detektion der Congestion im Router

- Paketverluste durch Buffer Overflow oder Time-out, Anzahl Retransmissions, Länge von Warteschlangen, Laufzeit und Varianz der Laufzeit
- Wahl eines sinnvollen Schwellwertes

Verbreitung der Information vom Router (nur Closed Loop)

- Senden dedizierter Pakete, erzeugt zusätzliche Last
- Setzen entsprechender Felder in ausgehenden Paketen
- Periodische Abfrage des Zustandes

Reaktion des Systems

- Reduktion der gesendeten Datenmenge eventuell mit Priorisierung unterschiedlicher Inhalte
- Erhöhen der Kapazität z.B. durch Backup-Router

# Beispiel: Messung der Auslastung

Beispiel für Messung der Auslastung im Router

- Ein Variable $u$ soll die Auslastung eines Ausgangsports eines Routers beschrieben, $0 \leq u \leq 1$.
- Für kurze Intervalle $i$ wird die Benutzung $b(i)$ binär gemessen, $b(i) \in \{0, 1\}$.
- Gewichtete Mittelung der Auslastung $u$ mit $u(i) = a \cdot u(i-1) + (1-a) \cdot b(i)$
- $a$ beschreibt die Gewichtung der Vergangenheit

Wenn die Auslastung $u$ einen Schwellwert überschreitet wird eine Warnung erzeugt. Alternative können z.B. die Speicherauslastung oder die Länge der Warteschlange gemessen werden.

# Strategien zur Vermeidung von Congestion

Sicherungsschicht / Data Link Layer

- Retransmission Policy: Zu viele Retransmissions durch zu kurze Time-out Intervalle oder Go-Back-n ARQ erhöhen Netzlast
- Out-Of-Order Caching Policy: Verwerfen von Out-Of-Order Paketen erfordert Retransmissions, besser Selective-Repeat ARQ
- Acknowledgement Policy: Extra ACK Pakete erzeugen zusätzliche Last. Verzögerte ACKs (z.B. durch Piggybacking auf Pakete in Rückrichtung) können Retransmissions erzeugen
- Flow Control Policy: Genau Kontrolle der Senderate (z.B. durch kleine Windowsize) kann Congestion verhindern

# Strategien zur Vermeidung von Congestion

Vermittlungsschicht / Netzwerkschicht

- verbindungsorientierte oder verbindungslose Übertragung: Viele Congestion Control Algorithmen funktionieren nur mit verbindungsorientierter Übertragung.
- Packet Queueing and Service Policy: Anzahl an Warteschlangen und Priorisierung
- Packet Discard Policy: Auswahl der zu verwerfenden Pakete z.B. nach Priorität
- Routing Algorithmus: Routing um überlastet Knoten herum
- Packet Lifetime Management: Pakete nicht zu früh (Retransmissions) aber auch nicht zu spät (belegte Resourcen) verwerfen

# Strategien zur Vermeidung von Congestion

Transportschicht

- Wie bei der Sicherungsschicht
  - Retransmission Policy
  - Out-Of-Order Caching Policy
  - Acknowledgement Policy
  - Flow Control Policy
- Timeout Policy: Große Laufzeitunterschiede durch das Netz im Vergleich zur einzelnen Verbindung zwischen zwei Routern

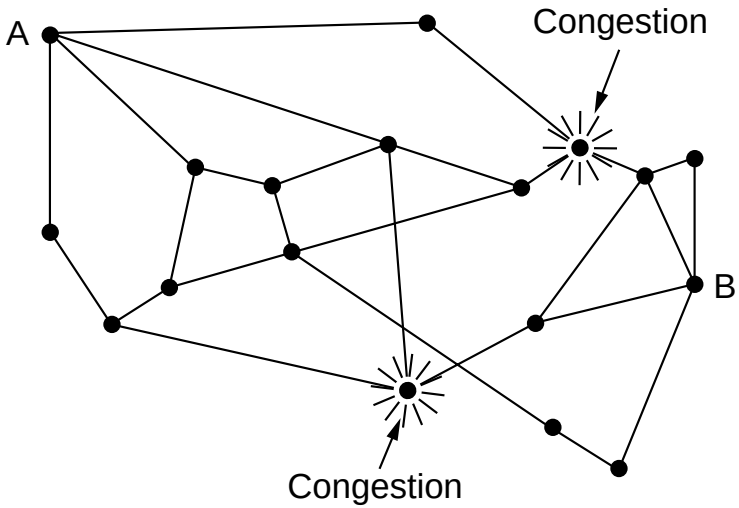
# Verbindungsorientierte Übertragung

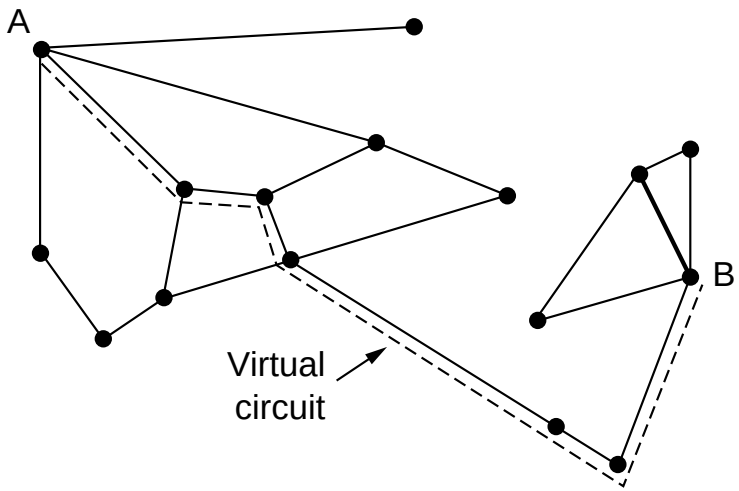
Bei verbindungsorientierter Übertragung ist in der Regel eine dynamische Reaktion auf eine Überlast möglich:

- Admission Control: Bei vorhandener Congestion werden keine neuen Verbindungen mehr zugelassen. Eventuell können dabei Prioritäten berücksichtigt werden.
- Resource Reservation: Beim Verbindungsaufbau werden die benötigten bzw. zur Verfügung gestellten Resourcen (Bandbreite, Speicher,...) abgestimmt. Diese Resource werden am Anfang oder nur bei detektierter Congestion reserviert.
- Alternative Routing: Beim Routing während des Verbindungsaufbaus werden Knoten mit bekannter Congestion vermieden.

# Verbindungsorientierte Übertragung

Alternative Routing:
Eine neue Verbindung (Virtual Circuit, VC) vermeidet bekannte Congestions.




(c) Tanenbaum, Computer Networks

# Verbindungslose Übertragung - Warning Bit

Warning Bit

- Relative altes Verfahren
- Bei Überlast setzt ein Router ein Flag (“Warning Bit”) im Header jedes Paketes.
- Wenn ein Paket mit Warning Bit am Empfänger ankommt wird dieses in das nächste Paket zum Sender (z.B. ein ACK) kopiert.
- Empfängt der Sender dieses Rückpaket reduziert er seine Senderate
- Nachteile
  - Unter Umständen große Verzögerung bei der Reaktion
  - Keine Information über den Ort der Congestion

# Verbindungslose Übertragung - Choke Packets

Choke Packets

- Bei Überlast sendet ein Router eine Meldung (“Choke Packet”) an den im Paket angegebenen Sender.
- Im Originalpaket wird ein Headerbit gesetzt, damit folgende Router keine Choke Packets mehr erzeugen.
- Empfängt der Sender das Choke Packet reduziert er seine Senderate
- Da bereits weitere Pakete unterwegs seien könnten, ignoriert der Sender weiter Choke Packets für dieses Ziel für ein bestimmtes Zeitintervall.
- Danach hört der Sender eine weiteres Intervall nach Choke Packets.
- Dann erhöht er ggf. seine Senderate wieder leicht.

# Verbindungslose Übertragung - Choke Packets

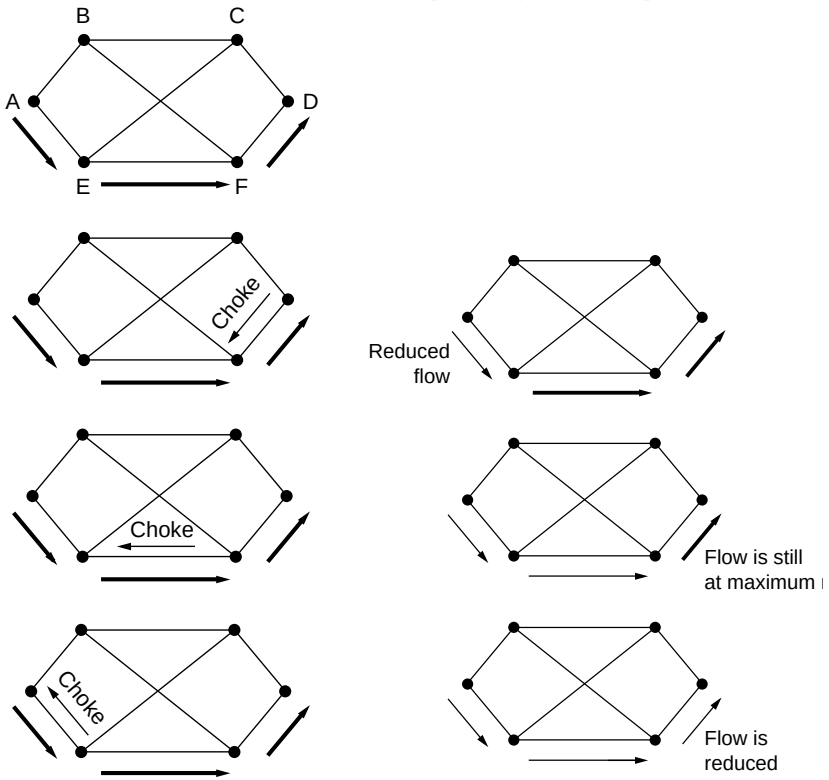
Hop-by-Hop Choke Packets

- Bei hohen Datenraten und langen Laufzeiten kann auch ein Choke Packet zu spät am Sender ankommen.
- Beim Hop-by-Hop Verfahren reagiert jeder Router auf Choke Packets und nicht nur der Sender.
- Empfängt ein Router ein Choke Packet reduziert auch er umgehende die Senderate.
- Erhöht den Speicherbedarf in Routern.
- Hop-by-Hop ermöglicht eine schnelle Reduktion der Datenrate am Punkt der Überlast.

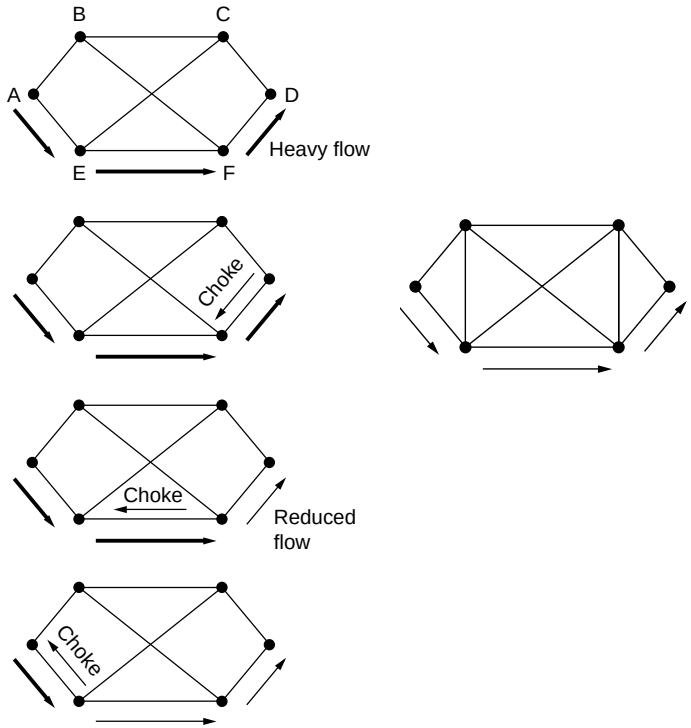
# Verbindungslose Übertragung - Choke Packets

Choke Packets ohne und mit Hop-by-Hop:

ohne Hop-by-Hop




mit Hop-by-Hop




(c) Tanenbaum, Computer Networks

# Load Shedding

Load Shedding (Lastabschaltung)

- Maßnahme bei andauernder Überlast
- Router wählen Pakete aus die verworfen werden.
- Intelligente Verfahren berücksichtigen z.B.
  - Prioritäten: Entweder auf den Dienst der gesamte Verbindung bezogen, z.B. E-Mail oder Video-Stream, oder in einem Dienst auf einzelne Pakete, z.B. die verschiedene MPEG-Rahmentypen.
  - Reihenfolge der Pakete: Sind alte Pakete wertvoller (z.B. bei Go-back-n ARQ) oder neue (z.B. bei Multimedia-Streaming)
  - Bei wenige Last dürfen umgekehrt Verbindungen mehr als die reservierte und garantierte Bandbreite benutzen.

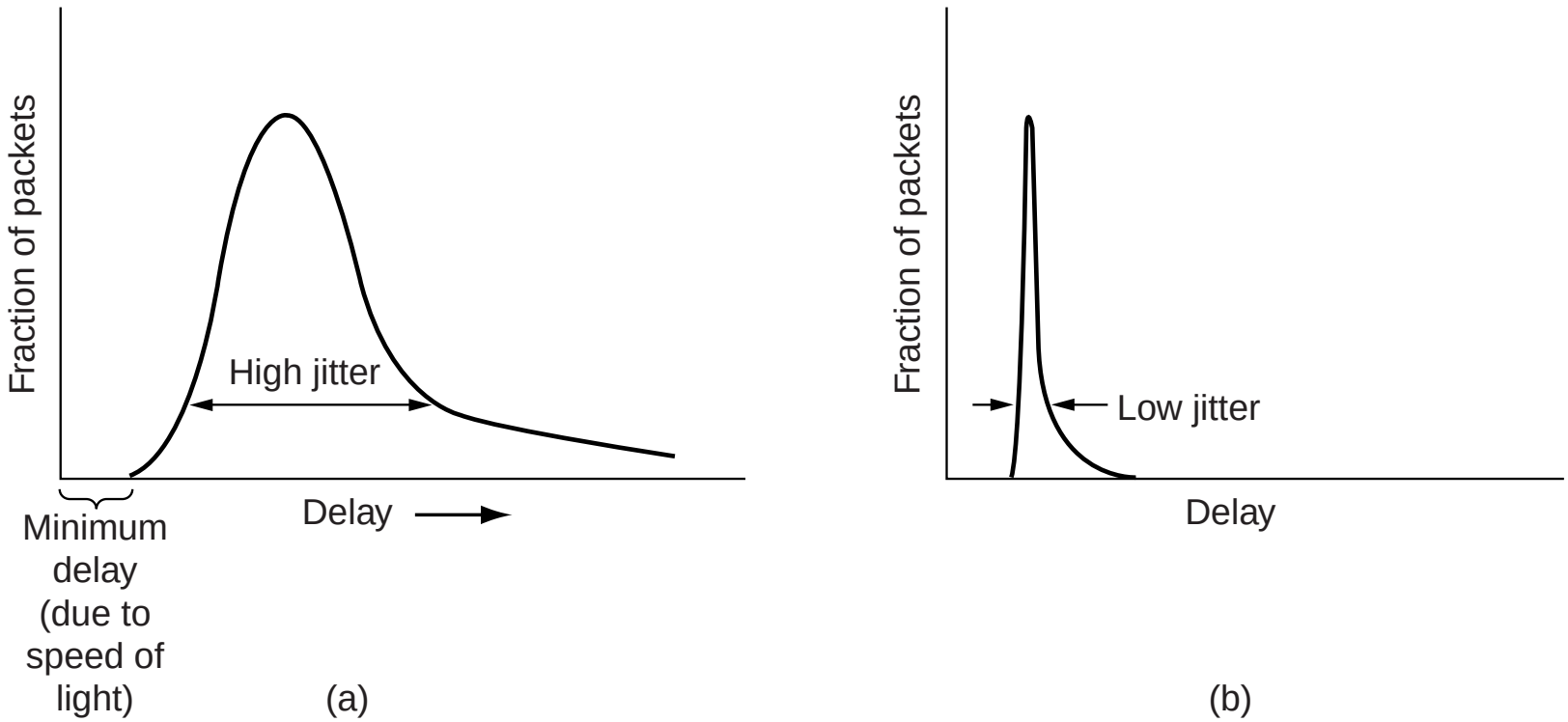
# Random Early Detection

Random Early Detection (RED) ist ein Versuch eine mögliche Überlast frühzeitig zu erkennen und gegenzusteuern.

- Einige Transportprotokolle (z.B. TCP) reagieren auf Paketverluste mit Reduktion der Senderate.
- Bei kabelgebundener Übertragung sind Übertragungsfehler unwahrscheinlich. In der Regel bedeuten Paketverluste Speicherüberlauf oder ähnliche Überlastung.
- Router wählen zufällig Pakete aus die verworfen werden. Es wird aber keine explizite Warnung geschickt.
- Der Sender detektiert den Paketverlust und seine Transportschicht reduziert die Datenrate

# Jitter Control

Jitter ist die Abweichung des Empfangszeitpunktes eines Paketes vom mittleren Empfangszeitpunkt aller Pakete.




(c) Tanenbaum, Computer Networks

# Jitter Control

Mögliche Jitter Control im Router

- der Router berechnet die erwartete Empfangszeit eines Paketes
- verspätete Pakete werden schnellstmöglich verarbeitet
- zu frühe Pakete werden verzögert
- der Router kann auch Jitter Control Information dem Paket hinzufügen

Streaming Applikationen erlauben in der Regel auch einen großen Buffer am Empfänger um Jitter zu kompensieren. Bei real-time Applikationen ist ein Buffer nicht möglich.

# Quality of Service (QoS)

# Einleitung

Quality of Service (QoS) beschreibt die Anforderungen einer Verbindung an die Dienstgüte. Dabei haben verschiedene Anwendungen unterschiedliche Anforderungen.

Die Kernparameter sind:

- Zuverlässigkeit (Reliability)
- Verzögerung/Laufzeit (Delay)
- Varianz der Laufzeit (Jitter)
- Bandbreite (Bandwidth)

Dabei müssen in der Regel auch Unterschiede zwischen verbindungsorientierter und verbindungsloser Übertragung berücksichtigt werden.

# Anforderungen von Anwendungen

| Application | Reliability | Delay | Jitter | Bandwidth |
|---|---|---|---|---|
| E-mail | High | Low | Low | Low |
| File transfer | High | Low | Low | Medium |
| Web access | High | Medium | Low | Medium |
| Remote login | High | Medium | Medium | Low |
| Audio on demand | Low | Low | High | Medium |
| Video on demand | Low | Low | High | High |
| Telephony | Low | High | High | Low |
| Videoconferencing | Low | High | High | High |

(c) Tanenbaum, Computer Networks

# Anforderungen von Anwendungen

Zuverlässigkeit (Reliability)

- High: Checksum und Acknowledgements benötigt
- Low: Paketfehler akzeptabel, d.h. keine Checksum und keine/weniger Acknowledgements

Verzögerung/Laufzeit (Delay)

- High: Real-time duplex Kommunikation, < 0.5s
- Medium: Interaktive Kommunikation, < 1-5s
- Low: Simplex Kommunikation, < 20s

# Anforderungen von Anwendungen

Varianz der Laufzeit (Jitter)

- High: interaktive und Audio/Video Kommunikation
- Low: nicht interaktive Kommunikation

Bandbreite (Bandwidth)

- High: Multimedia insbesondere Video Kommunikation
- Low: Text basierte Kommunikation

# QoS Klassen in ATM Netzwerken

In ATM (Asynchronous Transfer Mode) Netzen gibt es folgende QoS Klassen

- Konstante Bitrate (z.B. Telefonie)
- Real-time und variable Bitrate (z.B. Videokonferenz)
- Non-real-time und variable Bitrate (z.B. Video-on-Demand)
- verfügbare Bitrate (z.B. Dateitransfer)

Variable Bitrate entstehen z.B. bei Video Standards mit Kompression.

Diese Klassen lassen sich auch auf viele andere Netze übertragen.

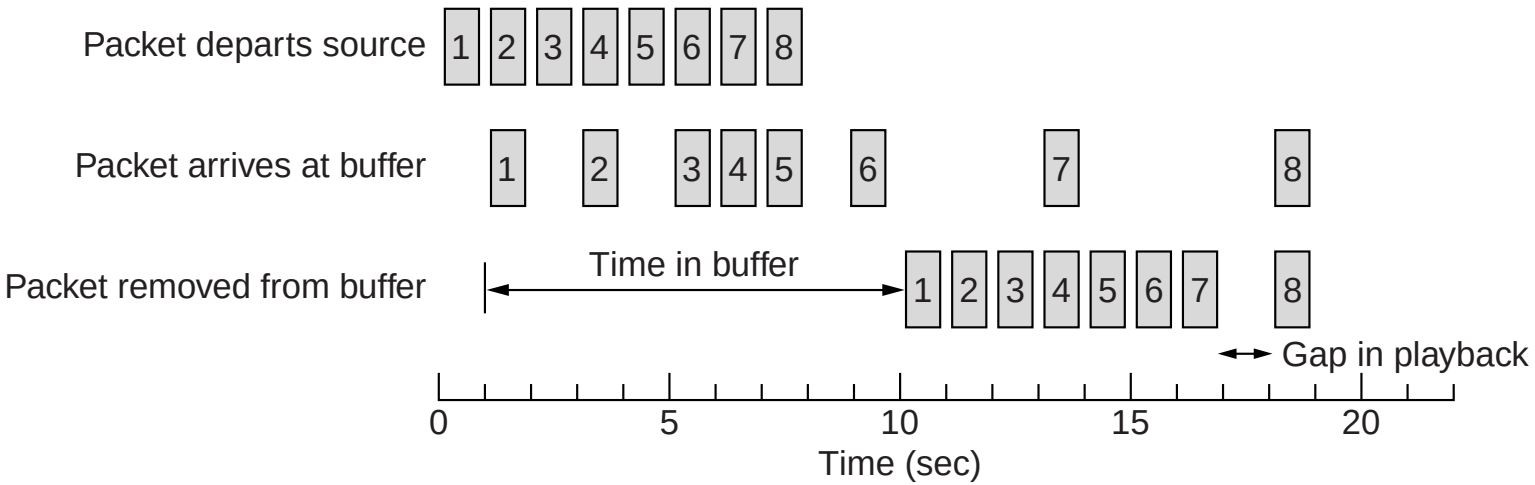
# Techniken für QoS

Keine Technik alleine liefert in der Regel eine akzeptable QoS unter den verschiedensten Szenarien. Es ist Kombination aus verschiedenen Techniken nötig.

**Overprovisioning:** Überdimensionierung des gesamten Systems bezüglich Bandbreite, Zwischenspeicher, Router Kapazität,... sodaß es nie Probleme gibt. Diese Technik ist natürlich sehr teuer und unter Umständen später sehr unflexibel.

# Buffering

Zum Beispiel in der Anwendungsschicht am Empfänger wird ein Speicher angelegt, der die unterschiedlich verzögerten Pakete zwischenspeichert und so den Jitter ausgleicht. Häufige Anwendung sind Audio-/Video-on-Demand.




(c) Tanenbaum, Computer Networks

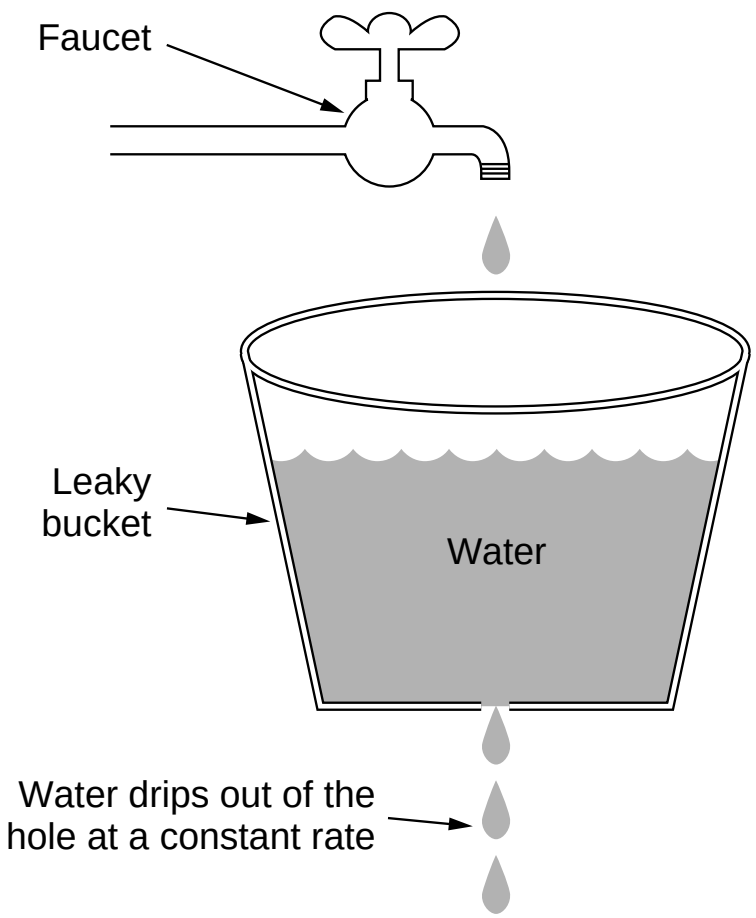
# Traffic Shaping

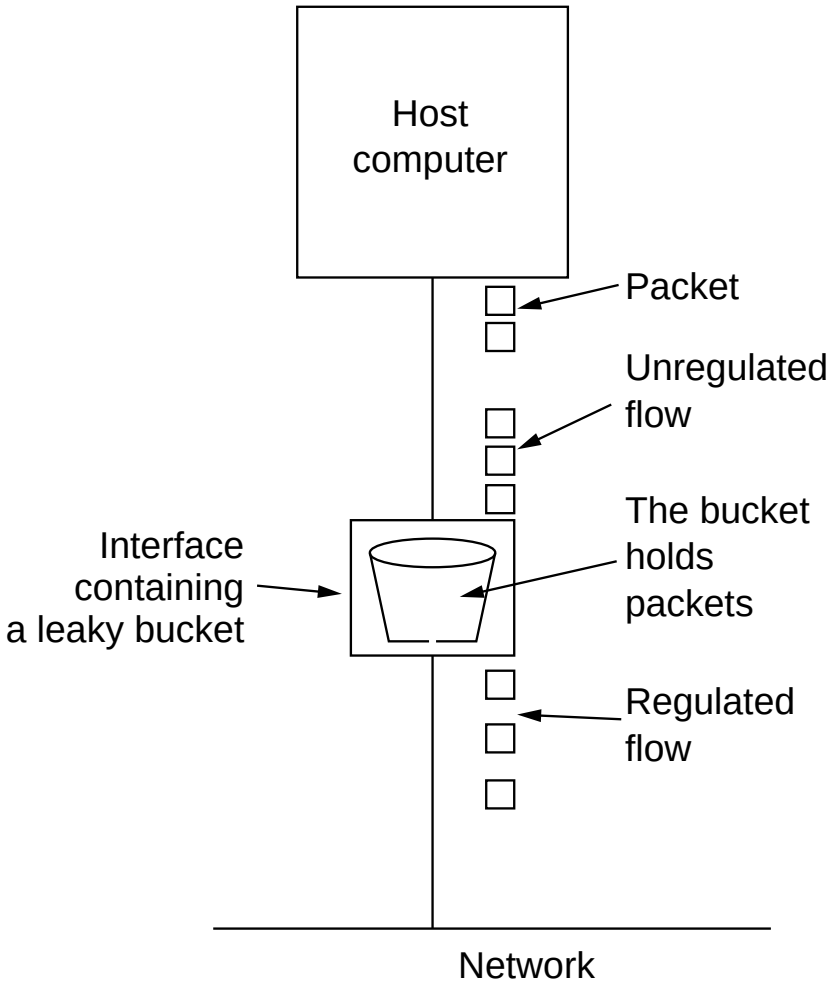
Selbst bei real-time Anwendungen sendet eine Server seine Daten oft in Bursts. Eine konstante Datenrate ist besser für die QoS.

Beim “Traffic Shaping” einigen sich Server/Sender und Netzwerk auf eine konstante mittlere Datenrate, die das Netzwerk garantiert überträgt (“service level agreement”). Der Sender/Server sendet nun mittels einem kleinen Zwischenspeicher die vereinbarten Datenrate.

Die Überwachung der Datenrate durch das Netzwerk wird “Traffic Policing” genannt.

# Leaky Bucket Algorithmus




(c) Tanenbaum, Computer Networks

# Leaky Bucket Algorithmus

Eine einfache Implementierung von Traffic Shaping ist der Leaky Bucket Algorithmus.

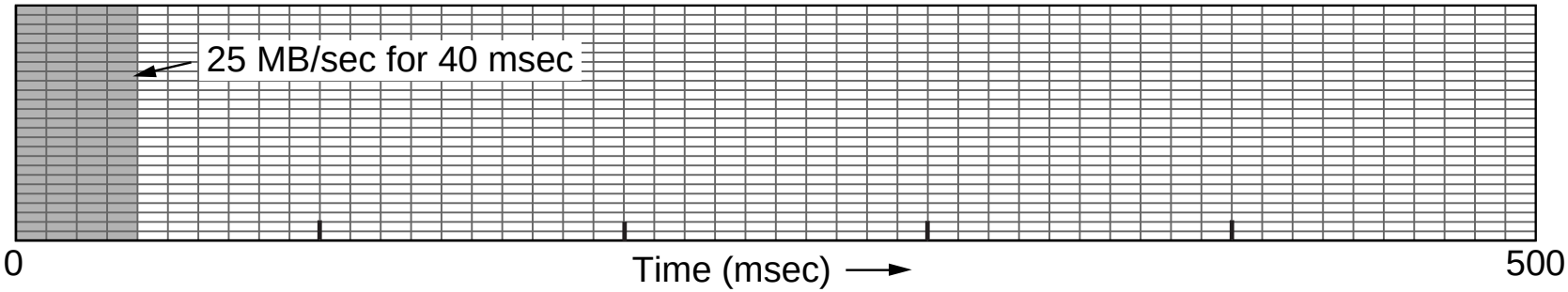
- Ein Interface im Sender implementiert eine Warteschlange, die am Ausgang eine konstante Anzahl an Paketen je Zeitintervall sendet.
- Bei verschieden großen Paketen kann es sinnvoller sein eine konstante Anzahl an Bytes je Zeitintervall zu senden.
- Ist die Warteschlange voll, werden eingehende Pakete verworfen.

Beispiel:

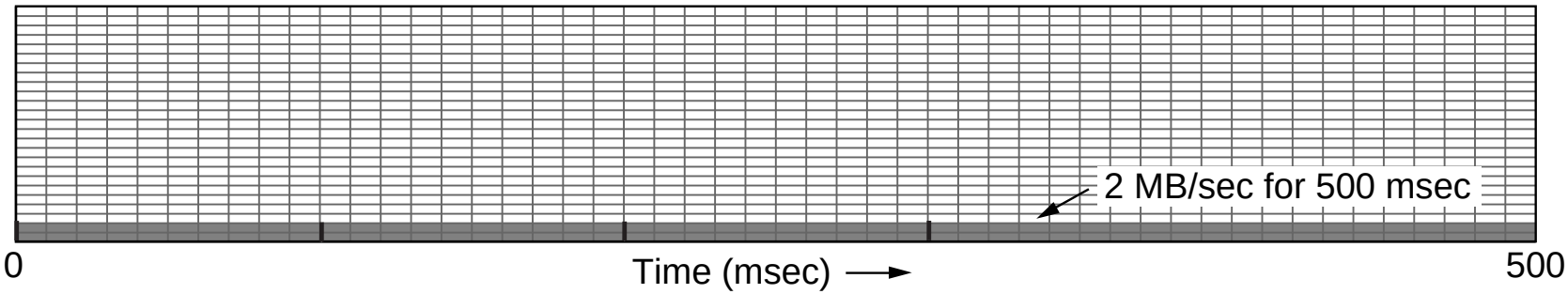
- Sender: Datenrate 25MB/s, Burstlänge 40ms, Burstintervall 1s, Burstgröße 1MB
- Router: kurzzeitig 25MB/s, Dauerbetrieb 2MB/s
- Leaky Bucket: Ausgangsrate 2MB/s, Kapazität 1MB

# Leaky Bucket Algorithmus - Beispiel

## Eingang Leaky Bucket:

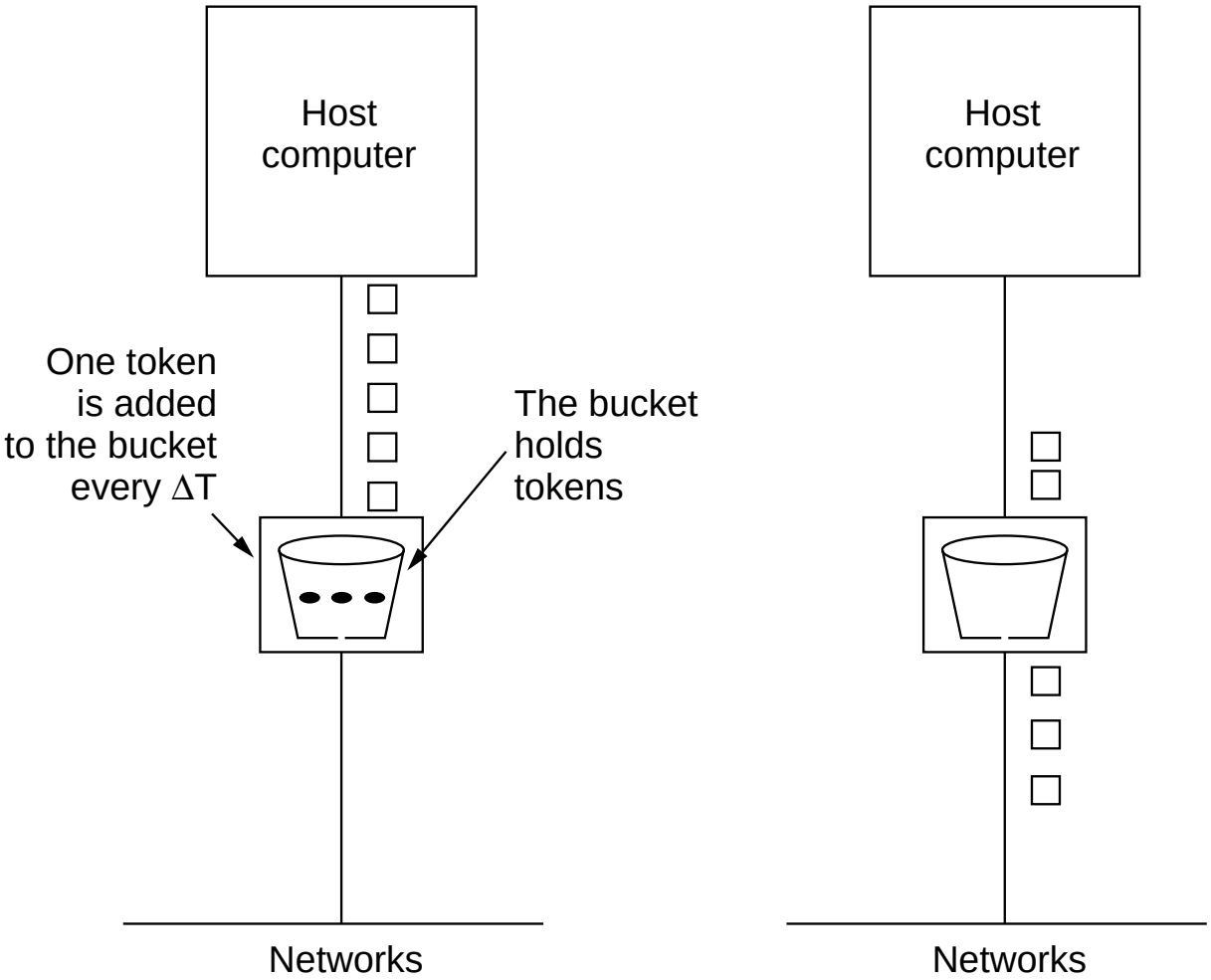



## Ausgang Leaky Bucket:




(c) Tanenbaum, Computer Networks

# Token Bucket Algorithmus




(c) Tanenbaum, Computer Networks

# Token Bucket Algorithmus

Der Leaky Bucket Algorithmus ist sehr unflexibel bei stark schwankenden Datenraten. Eine flexiblere Erweiterung ist der Token Bucket Algorithmus.

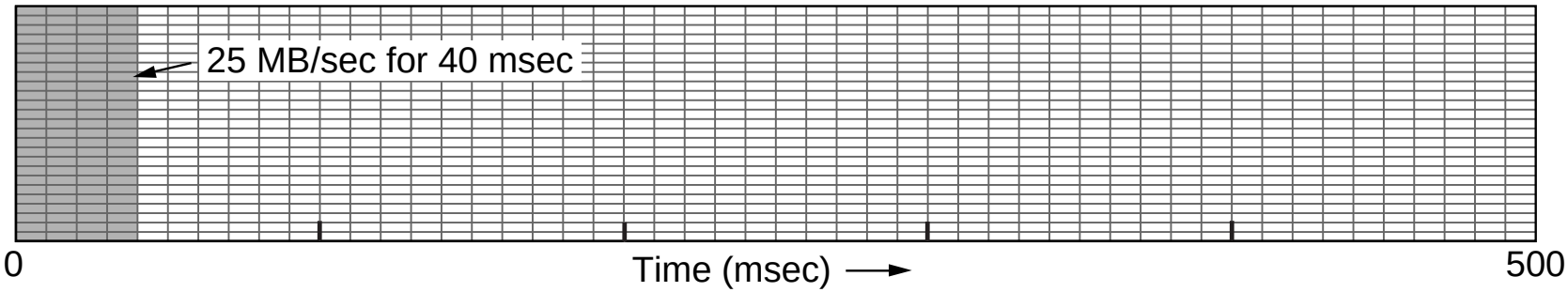
- Zusätzlich zur Warteschlange werden je Zeitintervall eine konstante Anzahl Tokens generiert.
- Jeder Token erlaubt die Übertragung einer bestimmten Anzahl an Paketen oder Bytes.
- Beim Senden werden die entsprechende Anzahl Token entfernt.
- Läuft die Tokenschlange über, werden neue Token verworfen. Pakete werden zunächst nicht verworfen.

Der Token Bucket Algorithmus kann Übertragungskapazität ansammeln um sie für Bursts zu verwenden.

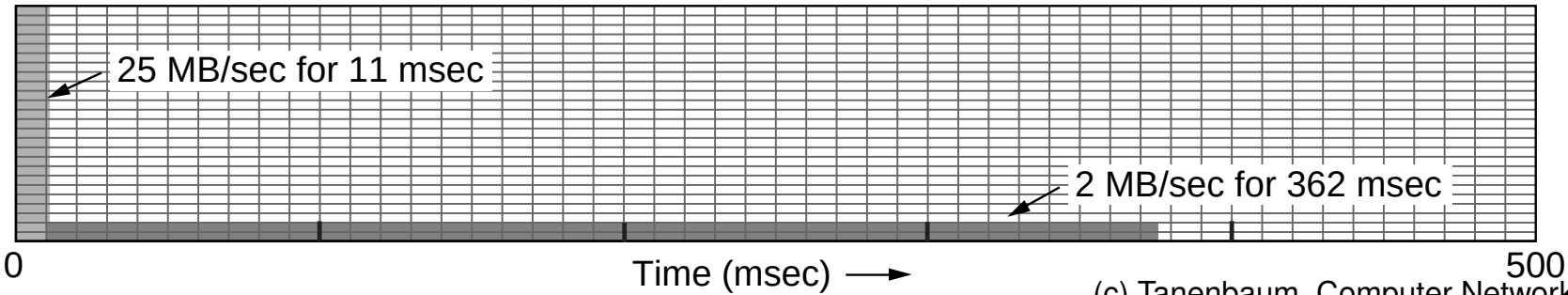
# Token Bucket Algorithmus - Beispiel

Token Bucket Kapazität: 250KB (initial gefüllt)
Token-Eingangsrate: 2MB/s

Eingang Token Bucket:




Ausgang Token Bucket:

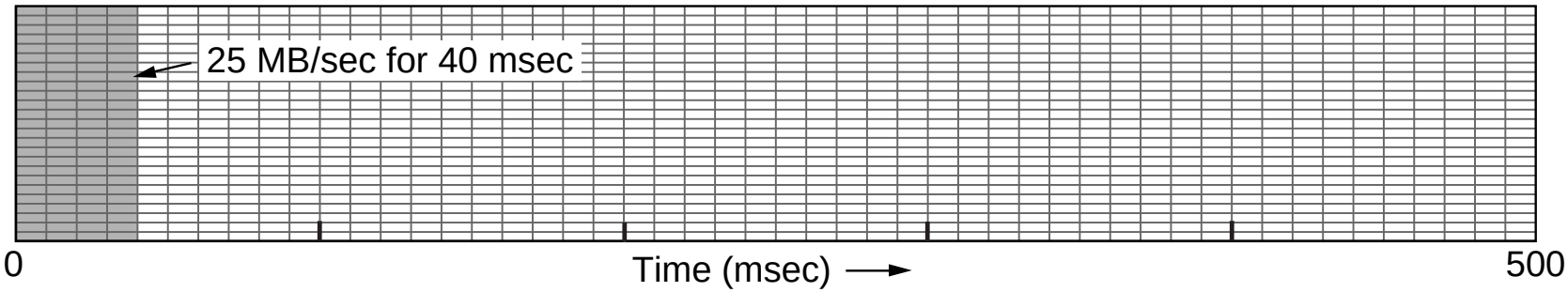



(c) Tanenbaum, Computer Networks

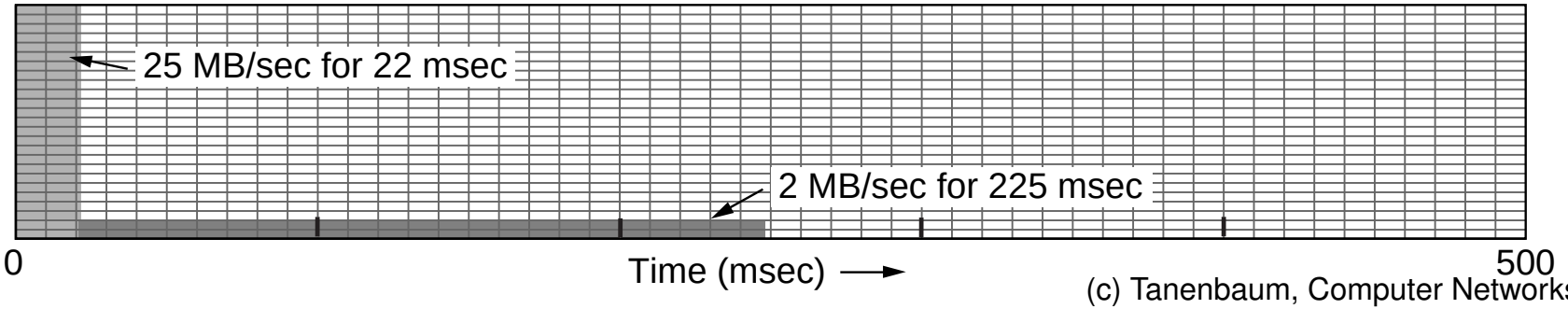
# Token Bucket Algorithmus - Beispiel

Token Bucket Kapazität: 500KB (initial gefüllt)
Token-Eingangsrate: 2MB/s

Eingang Token Bucket:




Ausgang Token Bucket:

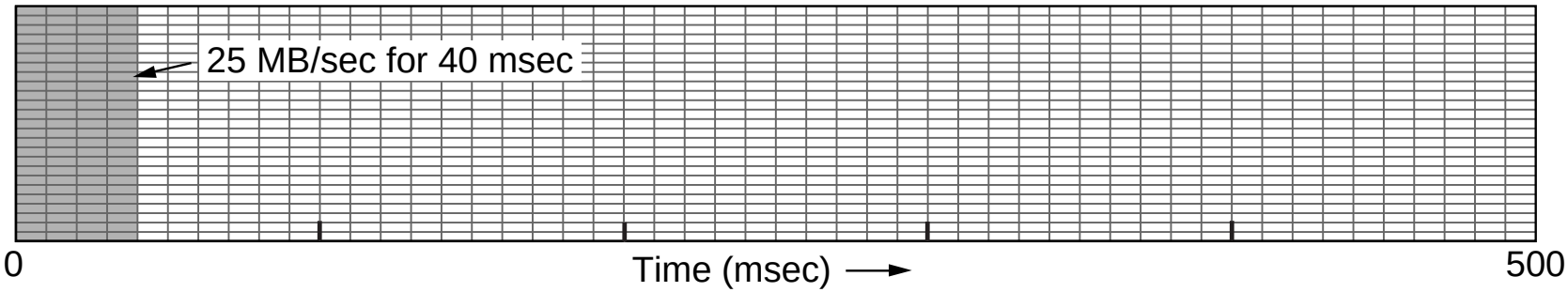



(c) Tanenbaum, Computer Networks

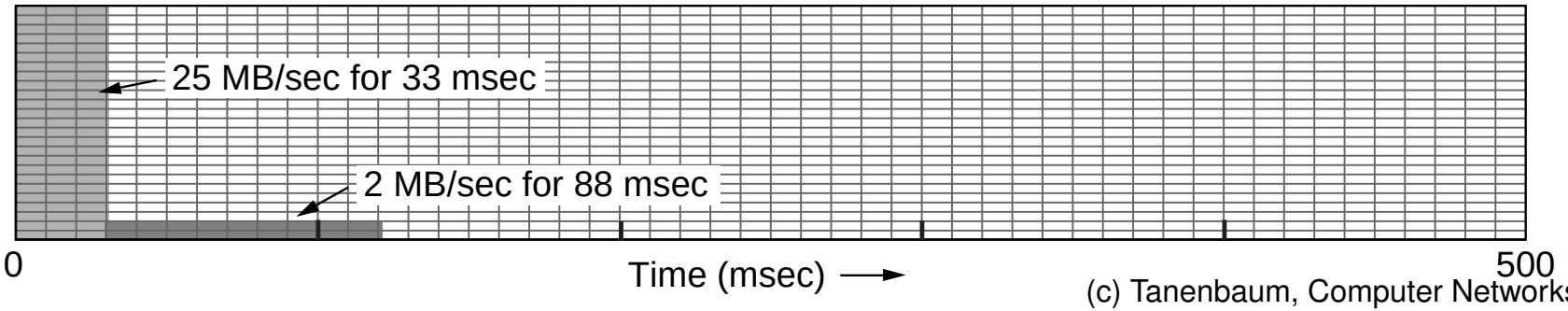
# Token Bucket Algorithmus - Beispiel

Token Bucket Kapazität: 750KB (initial gefüllt)
Token-Eingangsrate: 2MB/s

Eingang Token Bucket:




Ausgang Token Bucket:




(c) Tanenbaum, Computer Networks

# Token Bucket Algorithmus - Beispiel

Die Berechnung der Länge des Bursts am Ausgang mit maximaler Rate muß die während des Bursts eingehenden Token berücksichtigen.

Annahmen: Länge des Bursts am Ausgang $S$, Token Bucket Kapazität $C$ Bytes, Token-Eingangsrate $r$ in Bytes/s, maximale Ausgangsrate $M$ in Bytes/s.

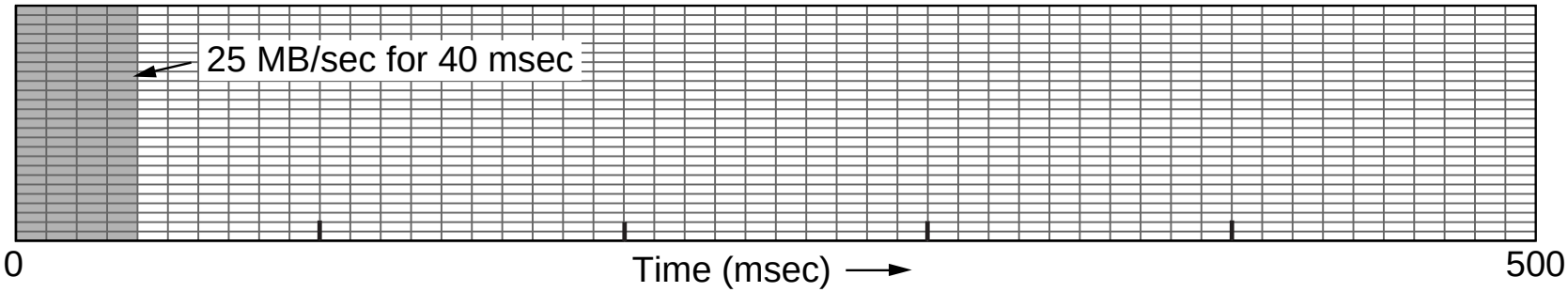
- Maximale Größe eines Burst am Ausgang: $C + rS$
- Datenmenge in einem Burst mit maximaler Rate: $MS$

$$\Rightarrow \quad C + rS = MS \quad \Rightarrow \quad S = C/(M - r)$$

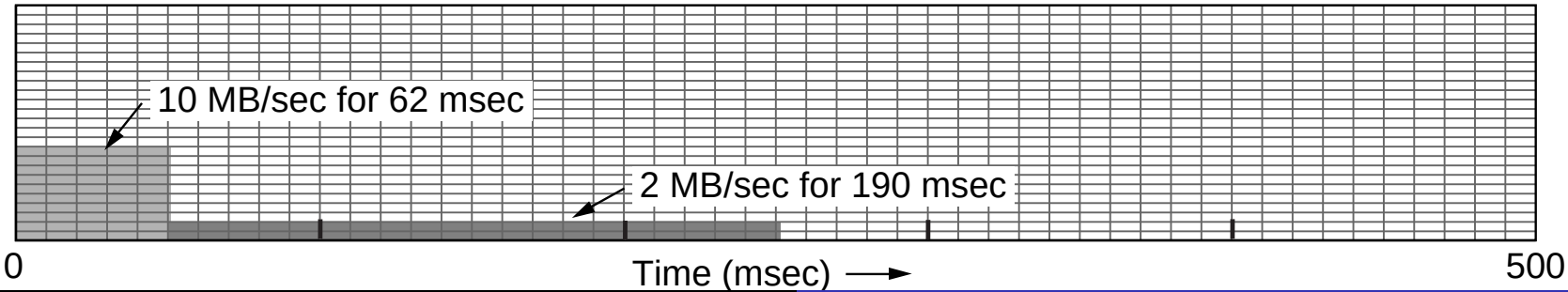
# Token + Leaky Bucket - Beispiel

Zur Vermeidung großer Bursts kann dem Token Bucket ein Leaky Bucket nachgeschaltet werden.
Token Bucket Kapazität: 750KB (gefüllt), Eingangsrate: 2MB/s
Leaky Bucket Ausgangsrate: 10MB/s

Eingang Token Bucket:




Ausgang Leaky Bucket:



# Resource Reservation

Um Resourcen effizient zu reservieren muß der Pfad der Daten bekannt sein. Dies ist einfach bei verbindungsorientierter Übertragung aber schwierig bei verbindungsloser Übertragung.

Die wichtigsten Resourcen sind

- Bandbreite
- Buffer Speicher
- CPU Zyklen

# Beispiel CPU Zyklen

CPU Auslastung und Verzögerung wird stark durch die statistische Verteilung der Aufgaben beeinflußt.

Beispiel:
Zufällige Ankunftszeiten der Pakete, im Mittel $\lambda$ Pakete/s
Zufällige CPU Last je Paket, im Mittel $\mu$ Pakete/s prozessierbar
Beide Zufallsprozesse seien Poisson verteilt.
Mittlere CPU Auslastung $\rho = \lambda/\mu$
Die mittlere Verzögerung $T$ für ein Paket ist dann

$$T = \frac{1}{\mu} \cdot \frac{1}{1-\lambda/\mu} = \frac{1}{\mu} \cdot \frac{1}{1-\rho}$$

Der Faktor $1/\mu$ ist die reine Rechenzeit ($\rho \approx 0 \Rightarrow T \approx 1/\mu$).
Der zweite Faktor beschreibt die Verzögerung durch die CPU Auslastung. Bei 50% Auslastung ergibt sich ein Faktor 2, bei 95% Auslastung ein Faktor 20.

# Admission Control

Mit der Admission Control wird entschieden ob eine neue Übertragung/Verbindung zugelassen wird, d.h. ob genug Resourcen vorhanden sind.

In der Regel erzeugt der Sender eine “flow specification” mit seinen Wünschen wie z.B.

| Parameter | Unit |
|---|---|
| Token bucket rate | Bytes/sec |
| Token bucket size | Bytes |
| Peak data rate | Bytes/sec |
| Minimum packet size | Bytes |
| Maximum packet size | Bytes |

# Admission Control

Dieser wird über die vereinbarte Route geschickt und jeder Router kann Werte dekrementieren wenn nötig. Der Empfänger sendet das Endergebnis zurück.

Wurde sich auf eine “flow specification” geeinigt, reservieren die Router die entsprechenden Resourcen.

Je exakter die Resourcen vorgegeben sind, desto einfacher ist es sie zu reservieren.

# Proportional Routing

Die meisten Routing Verfahren senden die Daten einer Verbindung über einen einzigen Pfad.

Für eine gute QoS kann es sinnvoll sein die Daten auf mehrere Pfade aufzuteilen.

- Verteilung der Bandbreite
- Erhöhung der Robustheit

Da die Router in der Regel nicht das ganze Netz kennen, können sie nur lokale Information verwenden. Beispielsweise können sie die Daten entsprechend der Kapazität der Ausgangsleitungen aufteilen.

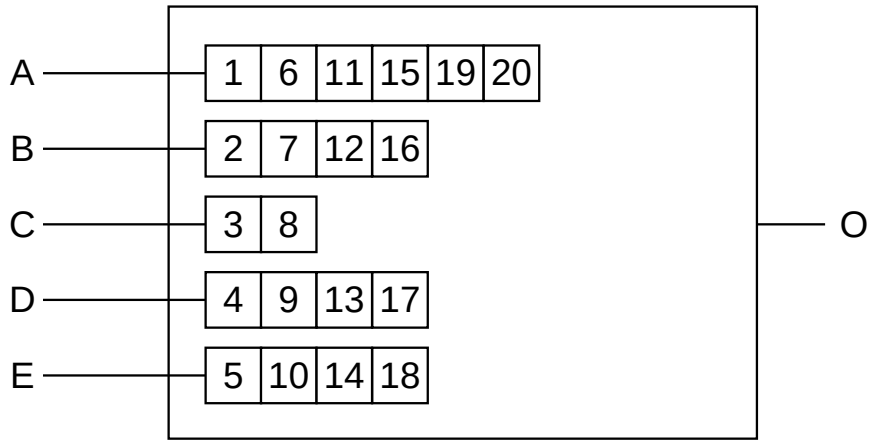
# Packet Scheduling

Wenn ein Router Pakete von mehreren Datenströmen bekommt, muß er diese in einer bestimmten Reihenfolge abarbeiten.

- FIFO:
  sehr einfach, bevorzugt aber aggressive Sender.
  Ungeeignet für QoS.
- Fair Queueing:
  Für jede Ausgangsleitung gibt es eine eigene Warteschlange. Diese wird round robin (packet-by-packet oder byte-by-byte) abgearbeitet.
- Weighted Fair Queueing:
  Erweiterung von Fair Queueing um eine Gewichtung der Datenströme entsprechend ihrer Priorität.

# Packet Scheduling

Beispiel für Fair Queueing mit byte-by-byte round robin:
a) Router mit 5 Eingängen A-E und einem Ausgang O
b) Endzeiten für die einzelnen Eingänge




(a)

| Packet | Finishing time |
|---|---|
| C | 8 |
| B | 16 |
| D | 17 |
| E | 18 |
| A | 20 |

(b)

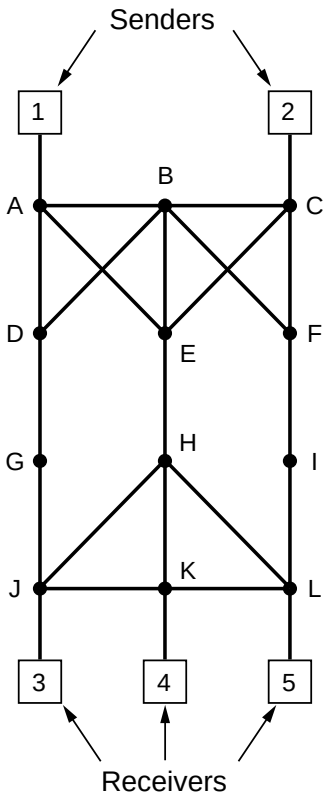
# Resource Reservation Protocol (RSVP)

RSVP ist das Kernprotokoll der “Integrated Services” Architektur von IETF. Integrated Services sind eine Architektur für Multicast (und Unicast) Multimedia Streaming.

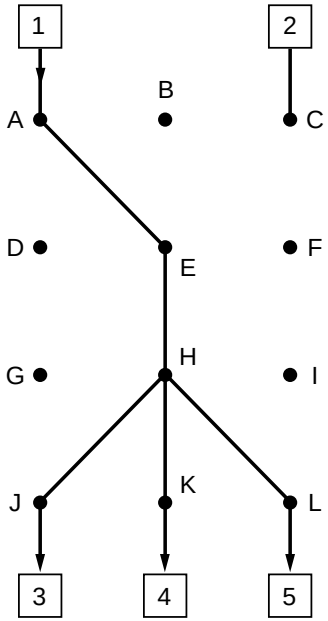
RSVP benutzt Multicast Routing, d.h. es werden Empfängergruppen gebildet und für jede Gruppe wird ein Routing Baum erzeugt. Bei RSVP wird noch zusätzlich Information per Multicast an die Router verteilt.

Jeder Empfänger kann eine “Reservation Message” an einen Sender schicken. Dieser wird mit Reverse Path Forwarding durch den Routing Baum propagiert. Dabei sieht sie jeder Router und reserviert entsprechende Resourcen. Falls keine Resourcen frei sind wird eine Fehlermeldung erzeugt. Datenströme von einem Sender werden zusammengefaßt. Dabei werden die höchsten QoS Anforderungen berücksichtigt.

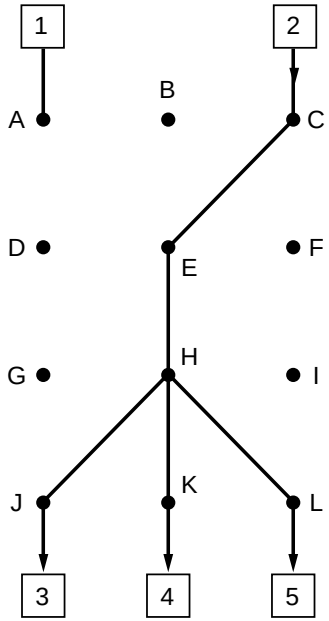
# RSVP Beispiel




(a)




(b)

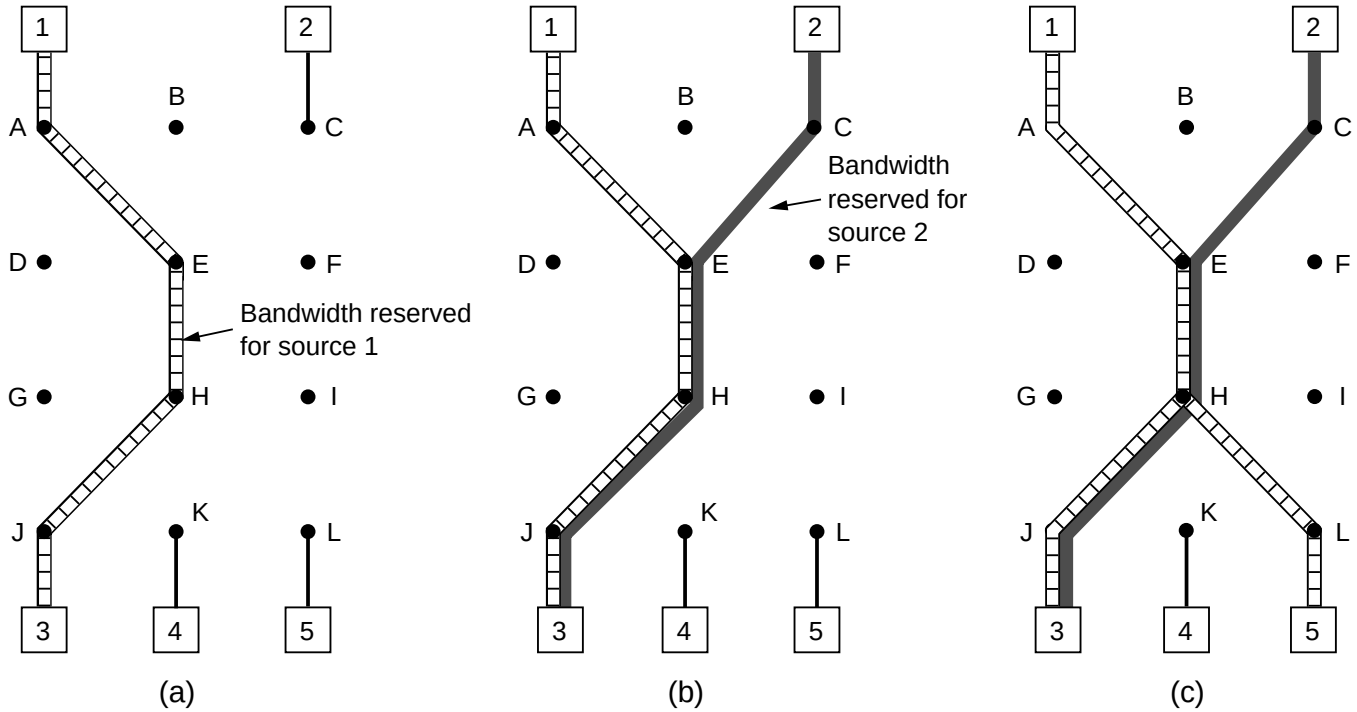



(c)



a) Netzwerk
b) Multicast Routing Baum Sender 1
b) Multicast Routing Baum Sender 2

# RSVP Beispiel




(c) Tanenbaum, Computer Networks

a) Empfänger 3 sendet Request für Sender 1
b) Empfänger 3 sendet Request für Sender 2
b) Empfänger 5 sendet Request für Sender 1

# Differentiated Services (DiffServ)

Protokolle wie RSVP offerieren einen guten QoS da sie die einzelnen Verbindung berücksichtigen. Sie sind aber relativ komplex und erfordern auch einen Vorlauf beim Einrichten der Verbindungen.

DiffServ sind eine einfache Alternative von IETF, die nicht konkrete Verbindungen berücksichtigt, sondern sich nur an Serviceklassen orientiert. DiffServ läßt sich lokal auf den Routern integrieren und erfordert keinen komplexen Verbindungsaufbau.

Einzelne Serviceklassen können z.B. durch das Type-of-Service Feld unterschieden werden. IETF versucht die Serviceklassen netzwerkunabhängig zu standardisieren.

# DiffServ - Expedited Forwarding

Expedited Forwarding ist das einfachste DiffServ Schema.

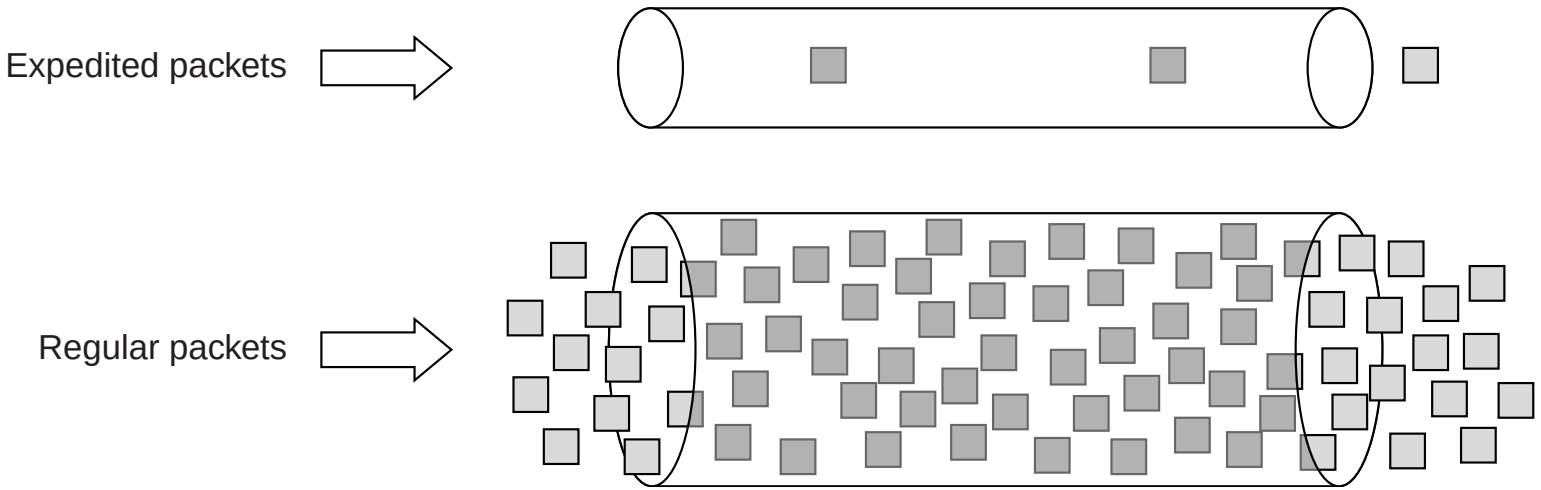
Es existieren zwei Serviceklassen:

- regular: Der Großteil der Pakete ohne besondere Anforderungen
- expedited: Wichtige Pakete, die ohne Verzögerung verarbeitet werden müssen

Für expedited Pakete soll das Netzwerk als leer erscheinen.

Eine mögliche Implementierung sind zwei virtuelle Kanäle für jede Verbindung. Entsprechend einer Art von Weighted Fair Queueing erhält der expedited Kanal dabei mehr Bandbreite als nötig zugewiesen.

# Expedited Forwarding - Beispiel




(c) Tanenbaum, Computer Networks

oben: Reservierter Kanal für expedited Pakete
unten: Kanal für reguläre Pakete

# DiffServ - Assured Forwarding

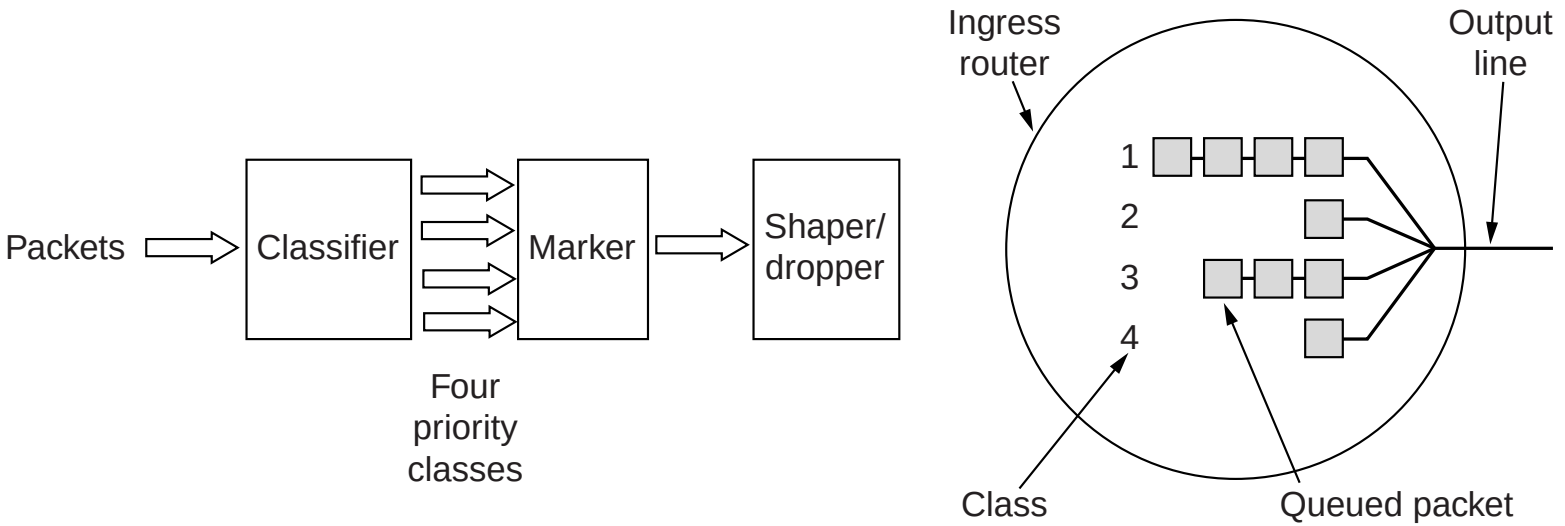
Ein weiteres DiffServ Schema ist Assured Forwarding.
Pakete werde klassifiziert nach

- Priority: 4 Klassen
- Discard Probability bei Congestion: 3 Klassen

D.h. es gibt insgesamt 12 Klassen.
Das Verfahren beinhaltet 3 Stufen:

- Klassifizierung nach Priorität z.B. im Sender oder im ersten Router
- Markieren der Pakete z.B. im Type-of-Service Feld
- Im Router werden dann in eine Warteschlangen-Filter entsprechend verzögert oder verworfen.

# Assured Forwarding - Beispiel




(c) Tanenbaum, Computer Networks