Faculty of Computer Science

universität wien

Netzwerktechnologie für Multimedia Anwendungen (NTM)

Kapitel 3

Florian Metzger
florian.metzger@univie.ac.at

David Stezenbach
david.stezenbach@univie.ac.at

Bachelorstudium Informatik
WS 2014/2015

3. Streaming

- 3.1 Netzwerkeigenschaften
- 3.2 Streaming Prinzipien
- 3.3 Protokolle
- 3.4 Voice over IP
- 3.5 Fehlertoleranz

3. Streaming

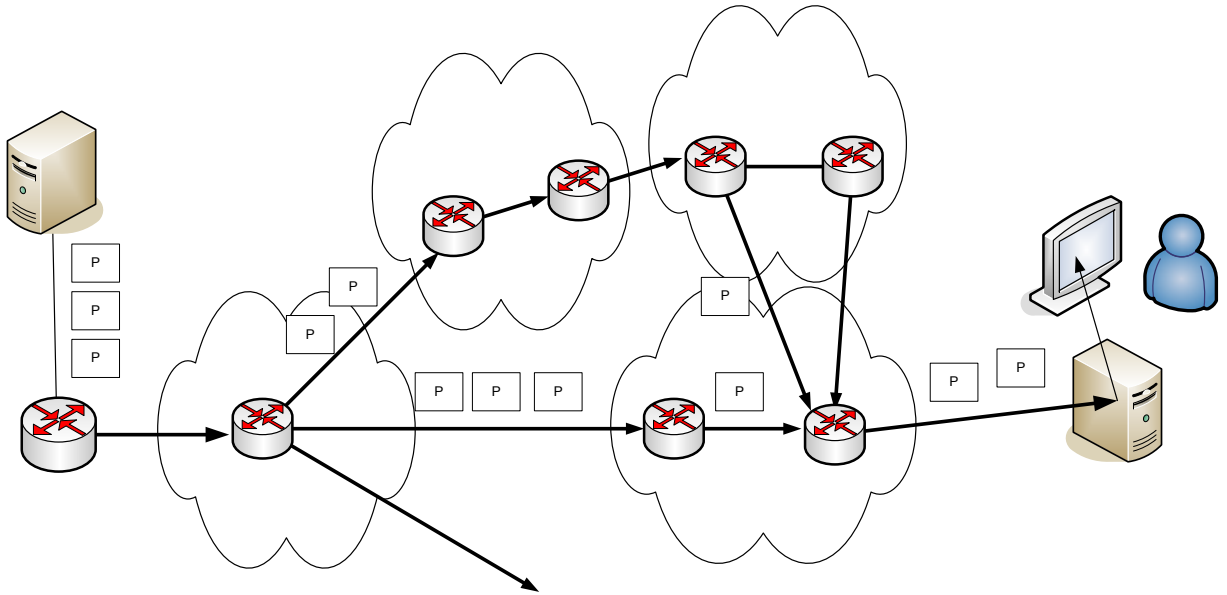
- **3.1 Netzwerkeigenschaften**
- 3.2 Streaming Prinzipien
- 3.3 Protokolle
- 3.4 Voice over IP
- 3.5 Fehlertoleranz

Netzwerke – Leitungsvermittelt

- Zwischen Sender und Empfänger wird ein Pfad geschaltet
  - TDM, FDM, SDM, …
  - Pfad-Ressourcen werden reserviert
  - z.B.: ISDN, PSTN, GSM, UMTS Circuit Switched, MPLS, ATM, usw.
- Konstante Latenz
- Erfordern Provider-übergreifendes Management (Pfad schalten), daher auch zentralisiertes QoS möglich
- Hoher Overhead, aufwendige Schnittstellen, Statusbehaftet
- Unflexibel, schlecht skalierbar, teuer in der Erhaltung

Netzwerke – Paketvermittelt

- Daten werden vom Sender in Pakete geteilt
  - Paketheader enthält Zieladresse
  - Pakete werden von Routern weitergeleitet
    - Zum jeweils nächsten Router
  - Beispiele: Internet/IP, GPRS/UMTS PS Domäne



Netzwerke – Paketvermittelt

- Erfordern keine Pfad-Schaltung
- Normalerweise keine Ende-zu-Ende QoS-Garantien (Netzneutralität)
- Zustandlose Router
- Flexibler und günstiger als Leitungsvermittlung

Unicast, Multicast, Broadcast

- Unicast:
  - Ein Sender, ein Empfänger
- Multicast:
  - Ein Sender
  - Eine Gruppe von Empfängern
  - Man muss sich in der Gruppe anmelden und Mitglied werden
- Broadcast:
  - Ein Sender, alle mit technischer Empfangseinrichtung können empfangen

Traffic Flow

Sequenz von Netzwerkpaketen/-daten

- Gehören zusammen
- Sind in einer eindeutigen Reihenfolge geordnet
- Sollten beim Empfänger in dieser Reihenfolge ankommen bzw. zurückgeordnet werden
- Definiert durch Quell- und Ziel-IP, Quell- und Ziel-Port und Transportprotokoll
- **Stream:** Besteht aus einem oder mehreren Flows

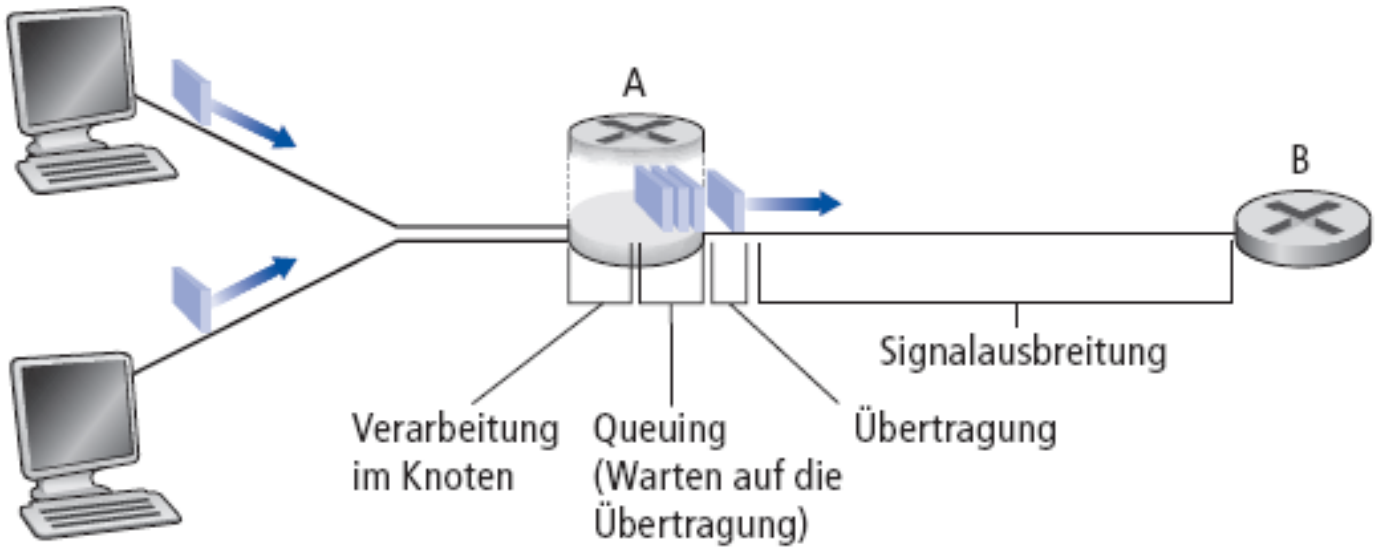
Quality of Service im Netzwerk

- Das Netzwerk soll die **Dienstgüte** liefern, welche für die jeweilige Anwendung notwendig ist.
- Typische Metriken
  - Bottleneck Bandwidth (Minimum der Bandbreiten eines Netzwerk-Pfades)
  - Throughput (Durchsatz der zwischen zwei Knoten gemessen wird)
  - Network Latency/Delay (Netzwerklatenz)
  - Jitter (Verzögerungsvarianz)
  - Packet loss (Paketverluste)
  - Per Flow Sequence Preservation (Wahrung der Paketreihenfolge)
  - Availability & Reliability (Verfügbarkeit & Zuverlässigkeit)

Netzwerk Latenzzeit

- Latency, One-Way Delay
  - Zeit, die ein Paket vom Sender zum Empfänger braucht
- Round Trip Time (RTT): Zeit hin und zurück
- Latenz in paketvermittelten Netzwerken
  - Je nach Synchronizität muss das Netzwerk die **Latenzzeit klein** halten
  - Wodurch werden Streaming-Pakete in einem paketorientierten Netzwerk verzögert?

Verzögerung, Verlust und Durchsatz in paketvermittelten Netzen

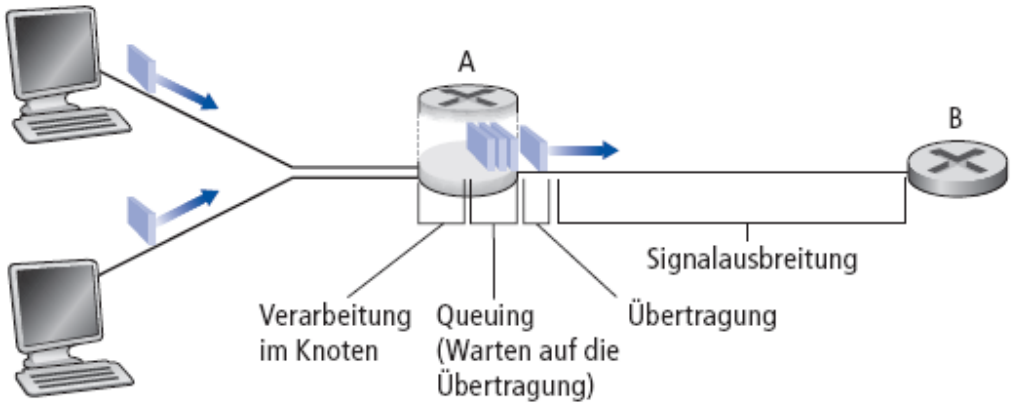


Wie entstehen Paketverluste und Verzögerungen?

→ Pakete warten in den Puffern von Routern wenn die Ankunftsrate die Kapazität der Ausgangsleitungen übersteigt

→ Ist die Warteschlange vor einer Leitung voll löscht der Router ankommende Pakete (da er keinen Platz hat um sie zu speichern), d.h. sie gehen verloren.

Arten der Verzögerung in Paketnetzen



1. Verarbeitung im Knoten:
   - Auf Bitfehler prüfen
   - Wahl der ausgehenden Leitung
2. Warten auf die Übertragung:
   - Wartezeit im Puffer, bis das Paket auf die Ausgangsleitung gelegt werden kann
   - Hängt von der Last auf der Ausgangsleitung ab
3. Übertragungsverzögerung:
   - Wenn R = Bandbreite einer Leitung (Bit/s) und L = Paketgröße (Bit), dann ist die Übertragungsverzögerung **= L/R**
4. Ausbreitungsverzögerung:
   - Wenn d = Länge der Leitung und s = Ausbreitungsgeschwindigkeit des Mediums (~2×10^8 m/s), dann ist die Ausbreitungsverzögerung **= d/s**

Verzögerung, Verlust und Durchsatz in paketvermittelten Netzen

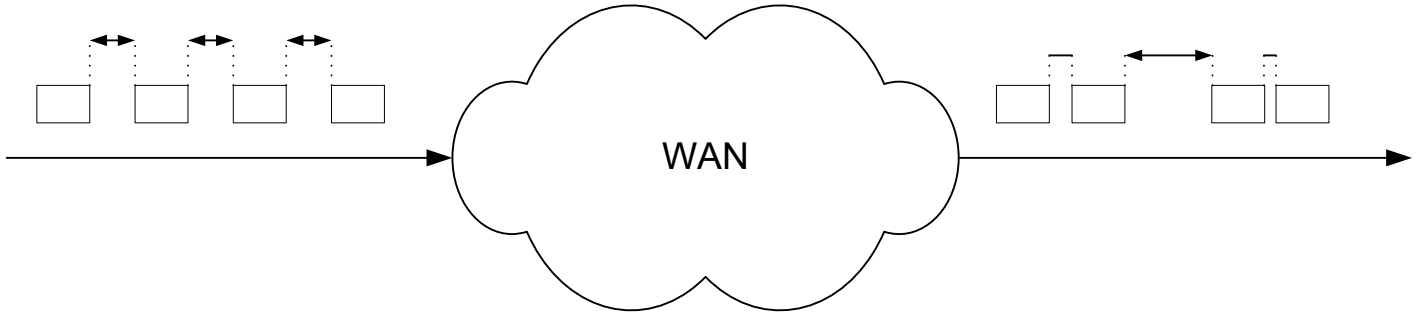
Gesamtverzögerung:

$$d_{\text{gesamt}} = d_{\text{Übertragung}} + d_{\text{Ausbreitung}} + d_{\text{Verarbeitung}} + d_{\text{Warten}}$$

- $d_{\text{Übertragung}}$ = Übertragungsverzögerung (Transmission Delay)
  - = L/R, signifikant wenn R klein ist
- $d_{\text{Ausbreitung}}$ = Ausbreitungsverzögerung (Propagation Delay)
  - Wenige Mikrosekunden bis einige hundert Millisekunden
- $d_{\text{Verarbeitung}}$ = Verarbeitungsverzögerung (Processing Delay)
  - Üblicherweise wenige Mikrosekunden oder weniger
- d_{Warten} = Wartezeit in Puffern (Queueing Delay)
  - Abhängig von der aktuellen Überlastsituation

Jitter

- Netzwerklatenz kann von Paket zu Paket schwanken
- Jitter ist die **Varianz** der Paket-Zwischenankunftszeiten (*Packet Interarrival Times*)



Verzögerung und Multimedia

- Pakete haben vorgesehene Abspielzeitpunkte
  - Ein Paket ist zu spät, wenn es zu dem Zeitpunkt, an dem es abgespielt werden soll, noch nicht angekommen und verfügbar ist
- Bei interaktiver Echtzeitkommunikation:
  - Ein Paket, das zu spät ankommt, gilt als verloren

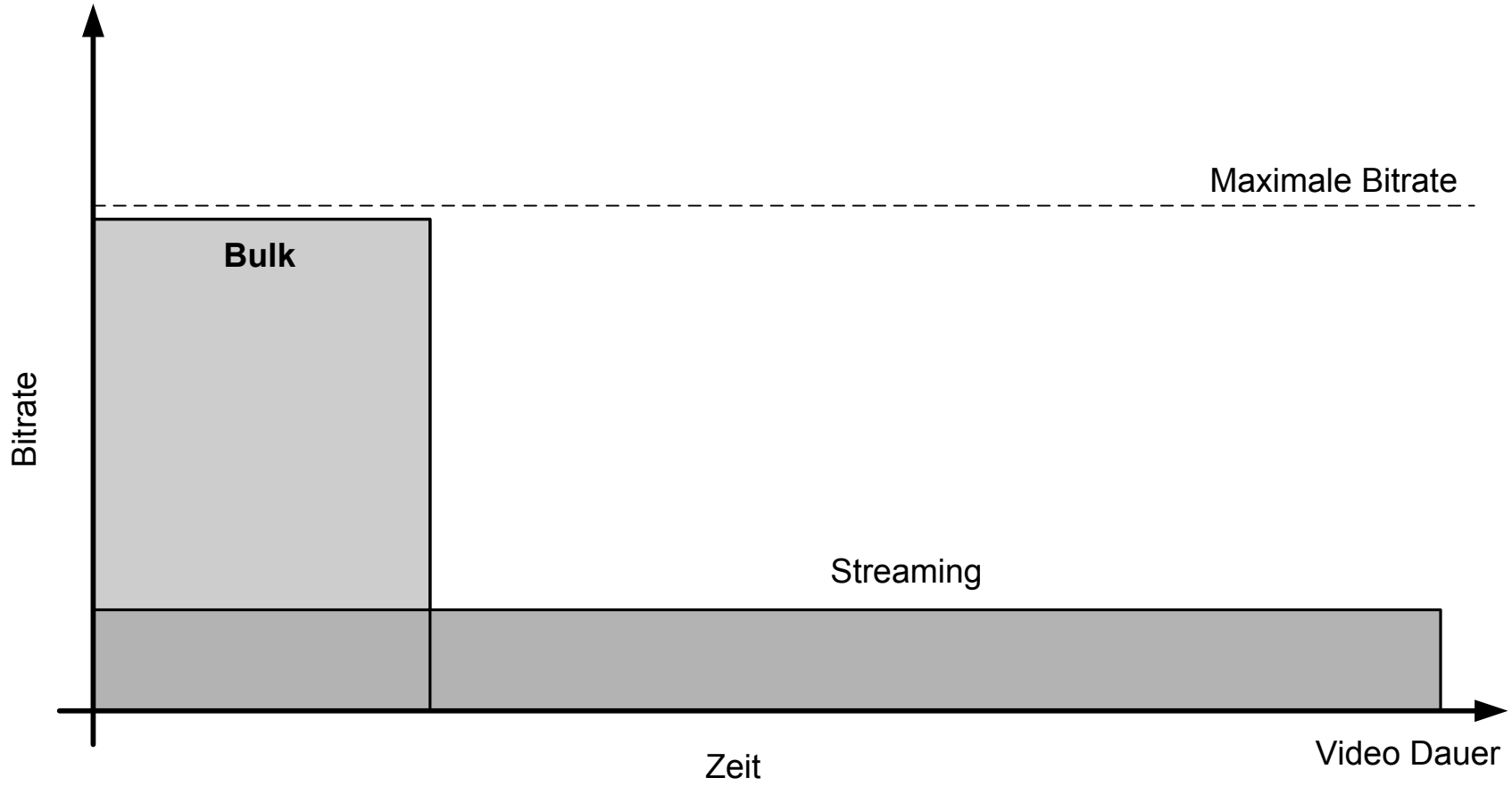
3. Streaming

- 3.1 Netzwerkeigenschaften
- **3.2 Streaming Prinzipien**
- 3.3 Protokolle
- 3.4 Voice over IP
- 3.5 Fehlertoleranz

Streaming – Anforderungen

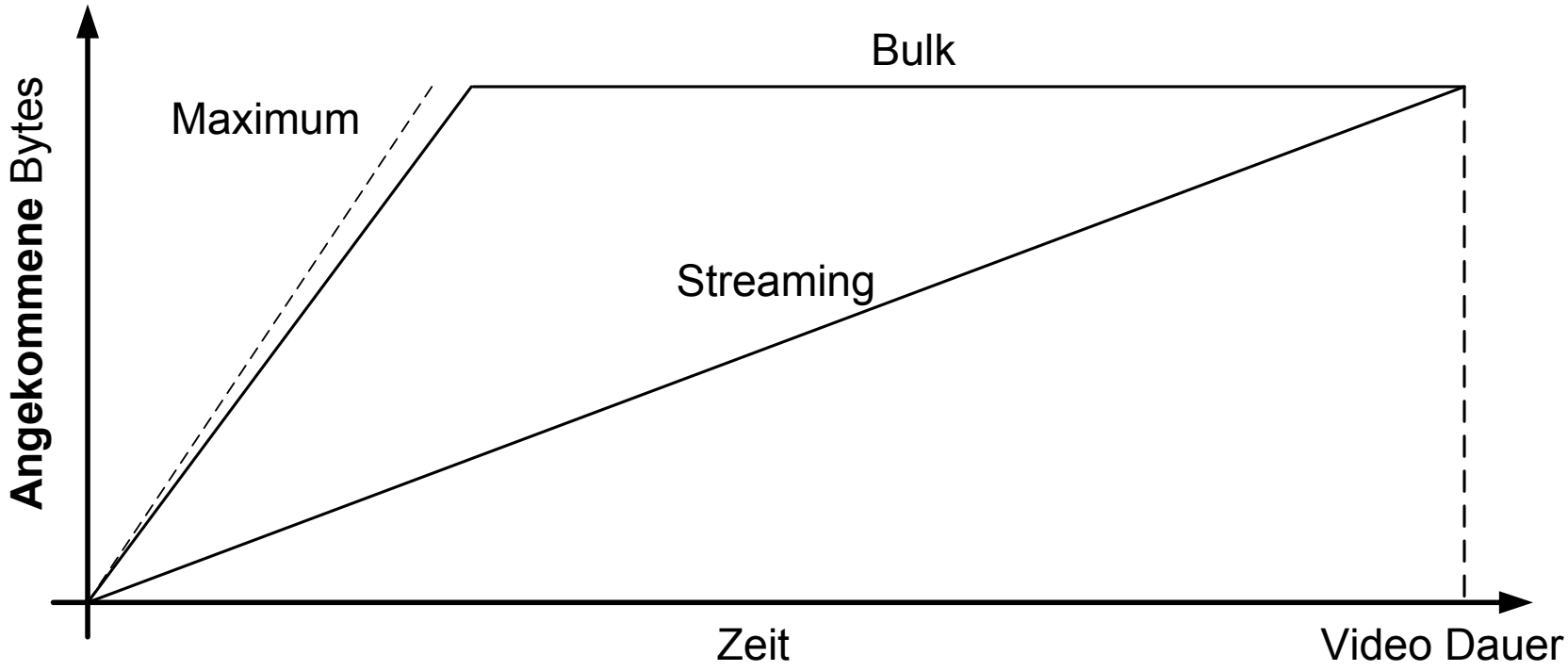
- Bulk Downloads
  - Eine bereits vorhandene Datei wird (mit TCP) mit der maximal möglichen Ende-zu-Ende Bandbreite verschickt
  - Hohe Elastizität
    - Geringe Anforderungen an QoS, Wartezeit
- Streaming
  - Übertragung **kontinuierlicher** Medien über paketvermittelte Netze
    - Präsentation sollte ungestört stattfinden
    - Zeitliche Abhängigkeiten zwischen den einzelnen Datenpaketen
    - Mediendaten müssen kontinuierlich beim Empfänger ankommen
  - Sender schickt Strom an Daten zum zeitnahen Abspielen
  - Ende-zu-Ende Bandbreite wird oft nicht vollständig belegt

Streaming vs. Bulk

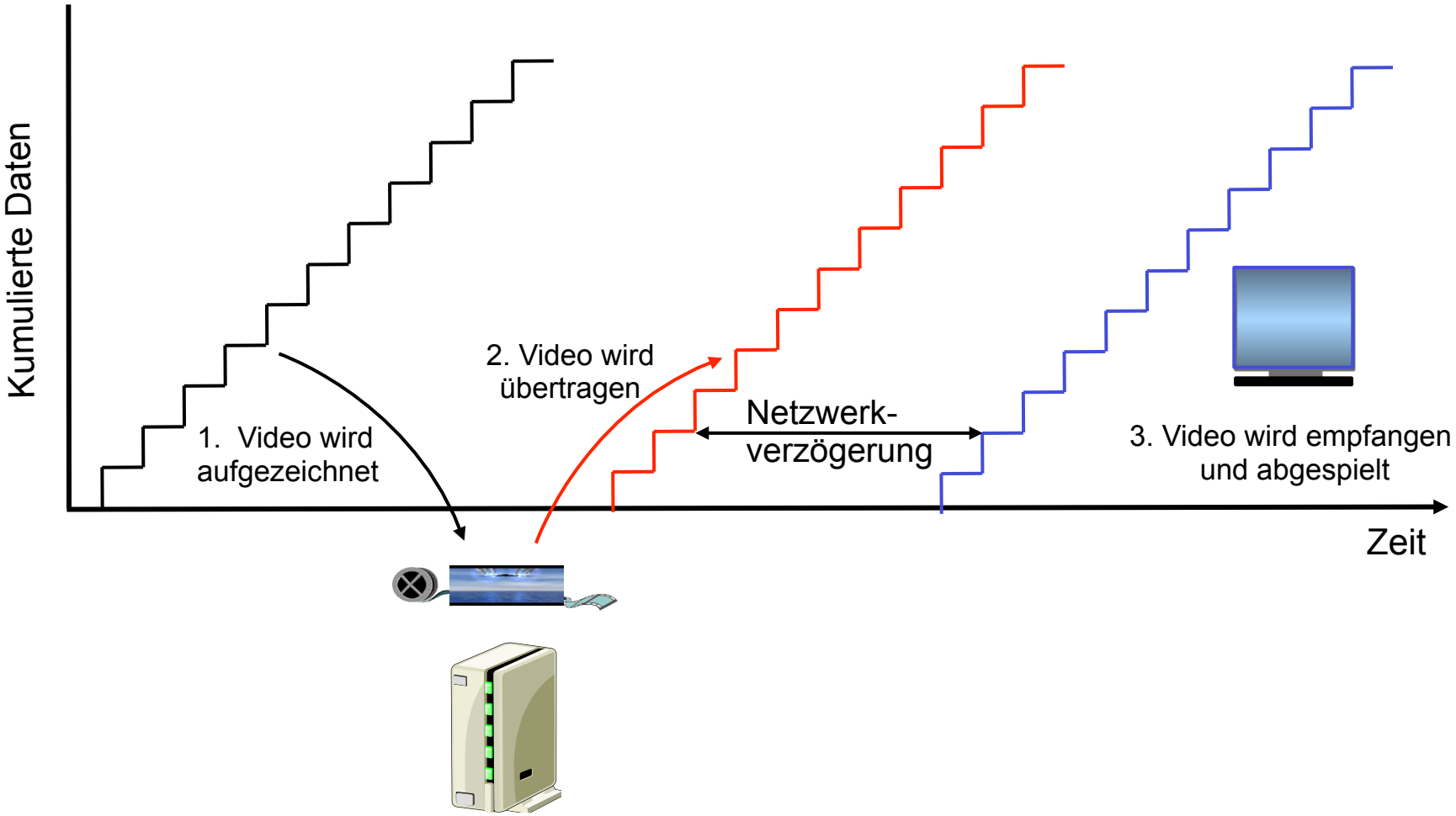


Streaming vs. Bulk

Wieviele Bytes sind bis zum Zeitpunkt t beim Empfänger angekommen ?



Streaming


Kumulierte Daten
1. Video wird aufgezeichnet
2. Video wird übertragen
Netzwerk-verzögerung
3. Video wird empfangen und abgespielt
Zeit

Streaming Klassifikationen

- Quelle/Art und “timeliness” des Inhalts
  - Stored, Live, Kommunikation
- Zuverlässigkeit des Transportsprotokolls
  - UDP, TCP (Retransmissions!)
- Ort der Kontrolle
  - Push- oder Pull-basiert
- Adaptivität der Inhalts-/Streamqualität
  - Implizit, Explizit
- Multiplexfähigkeit
  - Zahl der Empfänger eines einzelnen Streams
- Art der Übertragung

Streaming – Übertragung

- Synchronizität
  - Zeitliche Bindung entscheidet über die **notwendigen Garantien**, die das Netzwerk einhalten muss
  - Asynchrone, Synchrone, Isochrone Übertragungen
- Asynchrone Übertragung (Download)
  - *Bulk Transfer* (HTTP, …), Daten komplett in Puffer/Datei gespeichert
  - Präsentation erfolgt irgendwann nach Eintreffen des letzten Bytes
  - Benötigt großen lokaler Puffer/Speicher
  - Netzwerkdurchsatz bestimmt Wartezeit, Latenz und andere QoS-Parameter kaum relevant

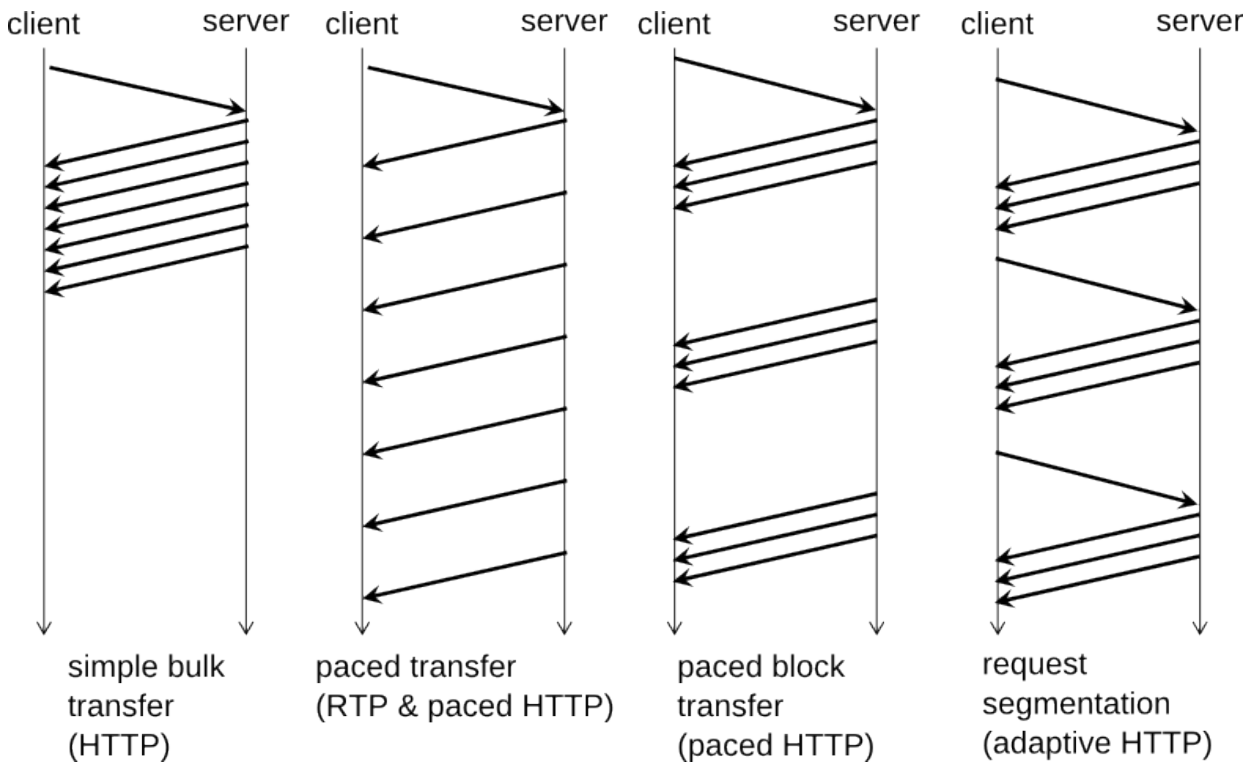
Streaming – Übertragung

- Synchrone Übertragung
  - Präsentation startet zeitversetzt aber noch während der Übertragung
  - *Progressive Download* braucht mittelgroßen Puffer
  - Abhängigkeit der Abspielqualität von der Netzwerkbandbreite
    - Falls Bandbreite kleiner als Video-Bitrate: Aussetzer, Wartezeiten (Stalling) oder verlorene Frames bzw. Teile von Frames
    - Qualität ist auch abhängig von der Abspielstrategie

Streaming – Übertragung

- Isochrone Übertragung
  - Interaktive Kommunikation
  - Präsentation erfolgt mit möglichst Kurzer Zeitverzögerung
  - Ziel: Verzögerung bis zum Abspielzeitpunkt möglichst klein halten aber dennoch möglichst gute Qualität erreichen
  - nur kleiner Puffer notwendig
  - Starker Einfluss durch Netzwerk-QoS
    - Bandbreite muss größer als Videobitrate sein oder Aussetzer
    - Einfluss durch hohe und schwankende Latenz und Paketverluste

Streaming – Übertragung



Einfluss des Transportprotokolls

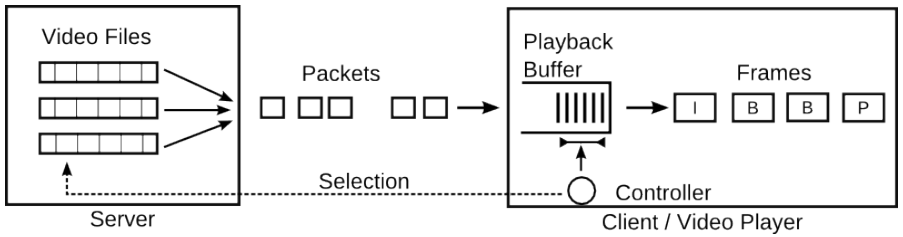
- Streaming sowohl mit UDP als auch TCP möglich
- UDP
  - Unzuverlässig, Fehlerbehandlung muss im Anwendungsprotokoll stattfinden
    - z.B. Übertragungswiederholungen falls es zeitlich Sinn macht
  - Paketverluste beeinflussen Bildqualität, implizit adaptiv
  - Passiert oft keine Firewalls, daher nur Einsatz in Provider-internen Netzen für Streaming
  - Senderate wird durch Anwendung bestimmt
    - Überlastgefahr oder Behandlung durch Anwendung nötig
  - Komplexere Anwendungen, aber auch mehr Kontrolle möglich

Einfluss des Transportprotokolls

- TCP
  - Durch Retransmissions keine Paketverluste für die Anwendung
  - Verlorene Pakete beeinflussen Durchsatz und Pufferfüllstand, es kann zu Stalling kommen
  - Generell etwas höhere Latenz durch Puffern und Abspielstrategien
  - Congestion Control und Equal Fair Share Bandwidth
  - Verwendung für interaktive Sprachkommunikation:
    - Kann Verlust einiger Pakete verschmerzen
    - Der Rest der Daten sollte aber nicht verzögert werden
    - Problematisch bei TCP durch Zuverlässigkeit
  - Kein Multicasting

Ort der Kontrolle & Abspielstrategien

- Serverseitig (Push)
  - Server entscheidet über Videoqualität, Übertragungsarten, …
  - Braucht Rückkanal, da Informationen nur beim Client vorhanden
- Clientseitig (Pull)
  - Client hat volle Kontrolle und holt sich die passenden Daten vom Videoserver
  - Braucht Abspielstrategie für Puffermanaget und zum Bestimmen der Abspielzeitpunkte
    - z.B.: mindestens 5s Video im Puffer, bevor mit dem Abspielen angefangen wird, um Pufferleerlauf zu vermeiden

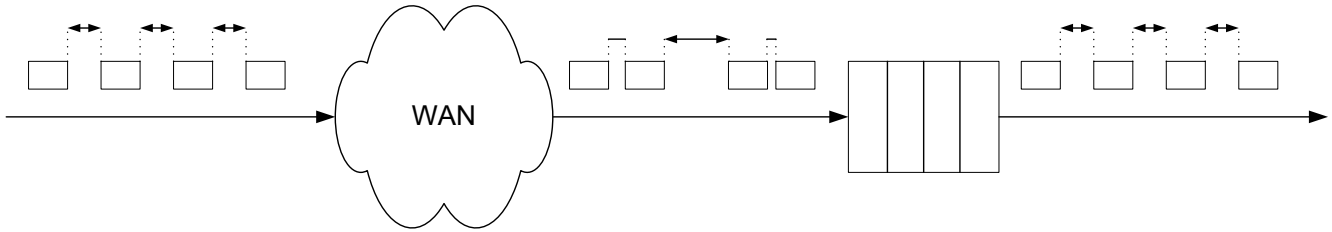


Streamingarten

- Video on Demand
  - Asynchron oder synchron
- Live Streaming
  - Synchron oder isochron
  - Zeitversatz klein
    - Ideal <1-2s
    - Zur besseren Lastverteilung (CDNs) auch ~60s üblich
- Kommunikation (“conversational”)
  - Normalerweise isochron
  - *ITU-T Recommendation G.114 for One-way Transmission Time*:
    - 0 – 150 ms: Für die meisten Zwecke ausreichend
    - 150 – 400 ms: Immer noch akzeptabel
    - > 400 ms: Inakzeptabel für Echtzeitanwendungen

Puffer beim Empfänger

- Pakete wandern beim **Empfänger** in eine Reihe von **Puffern** und werden dort verzögert
  - Jitter smoothing bei Constant Bit-Rate (CBR) Verkehr und bei VoIP
    - Jitter Puffer wirkt wie ein Traffic Shaper

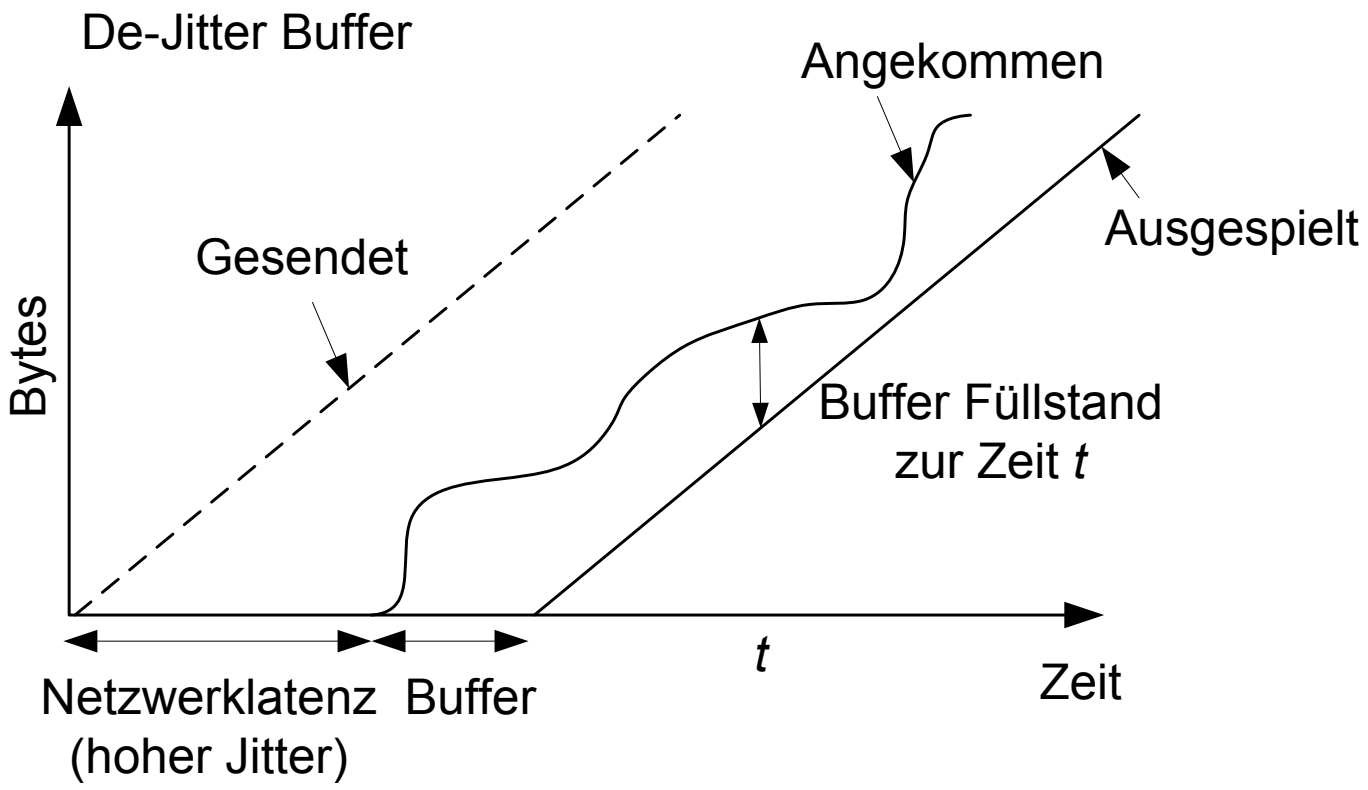


  - Inter-Stream Synchronisation
  - Audio und Video sollen synchronisiert werden (Lip Sync)
  - Re-request
    - Verlorene Pakete können erneut angefordert und noch einmal geschickt werden

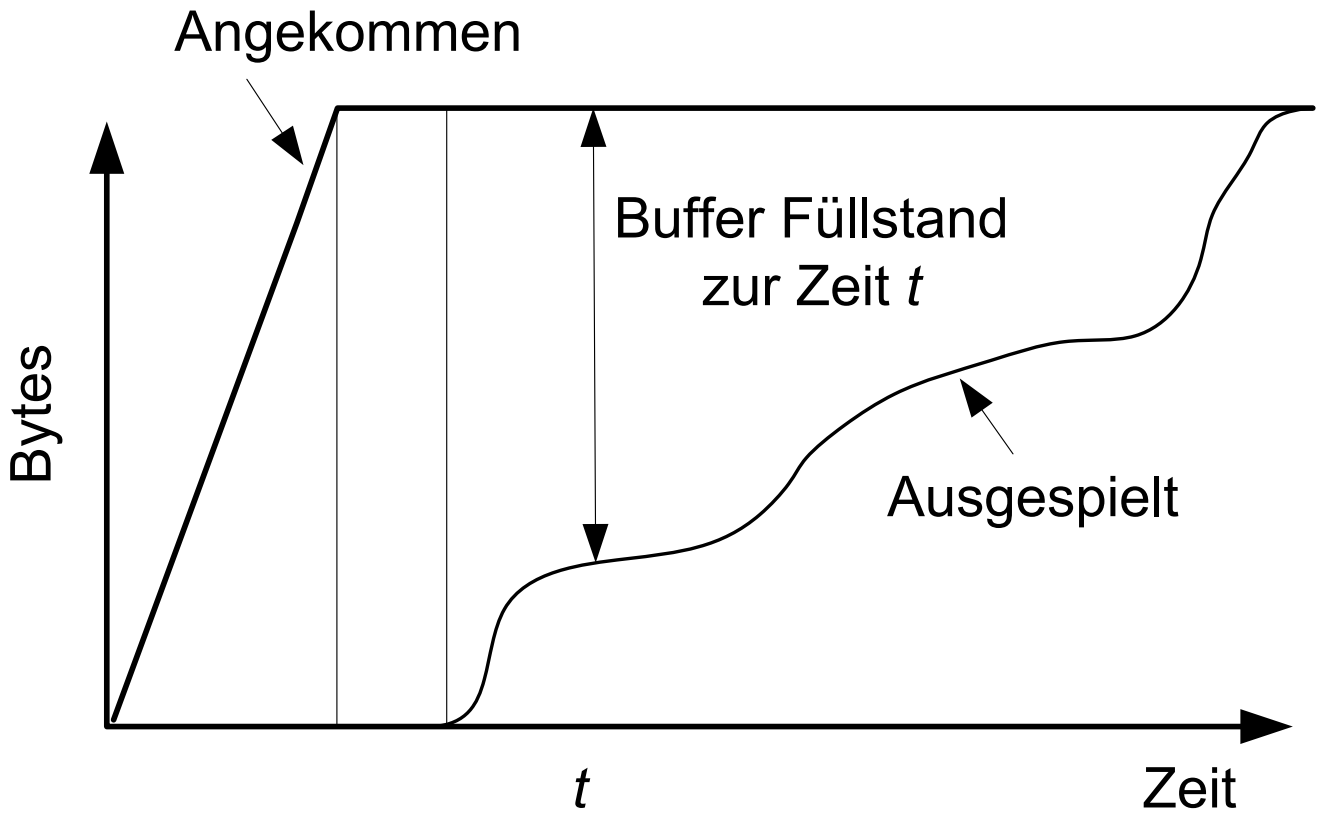
Puffer-Parameter

- Verzögerungszeit t
- Puffergröße b
- Verzögerungszeit im De-Jitter/Decoder Puffer
  - Kurz → nur geringer Jitter kann ausgeglichen werden
  - Lang → großer Jitter kann ausgeglichen werden
- Faustregel: Pufferverzögerung für De-Jitter
  $\Delta t = \Sigma q_i$ für q_i Queuing Delay des Knoten i
  - Beispiel: *VoIP Buffering typischerweise bis zu* 70-80 ms

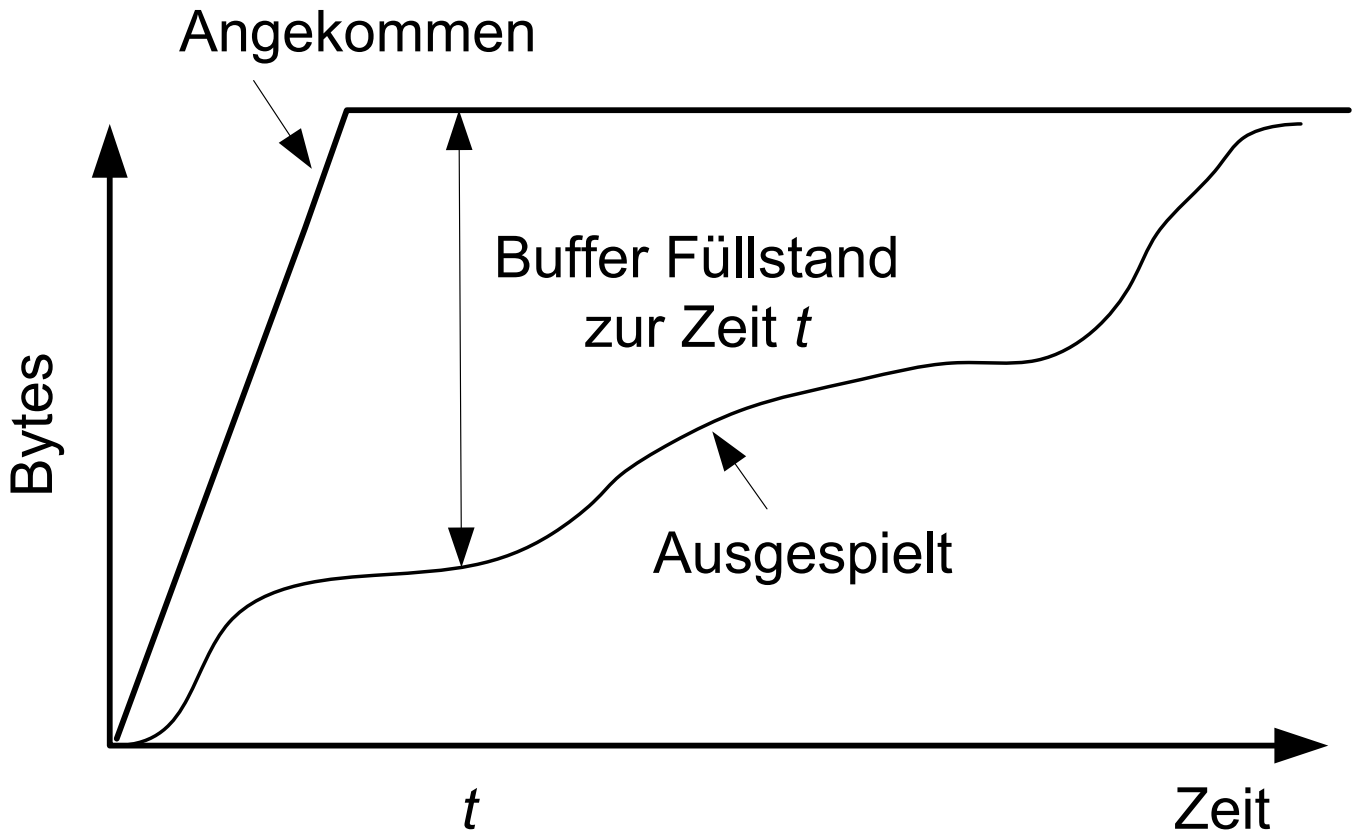
Puffer – CBR, Hoher Jitter



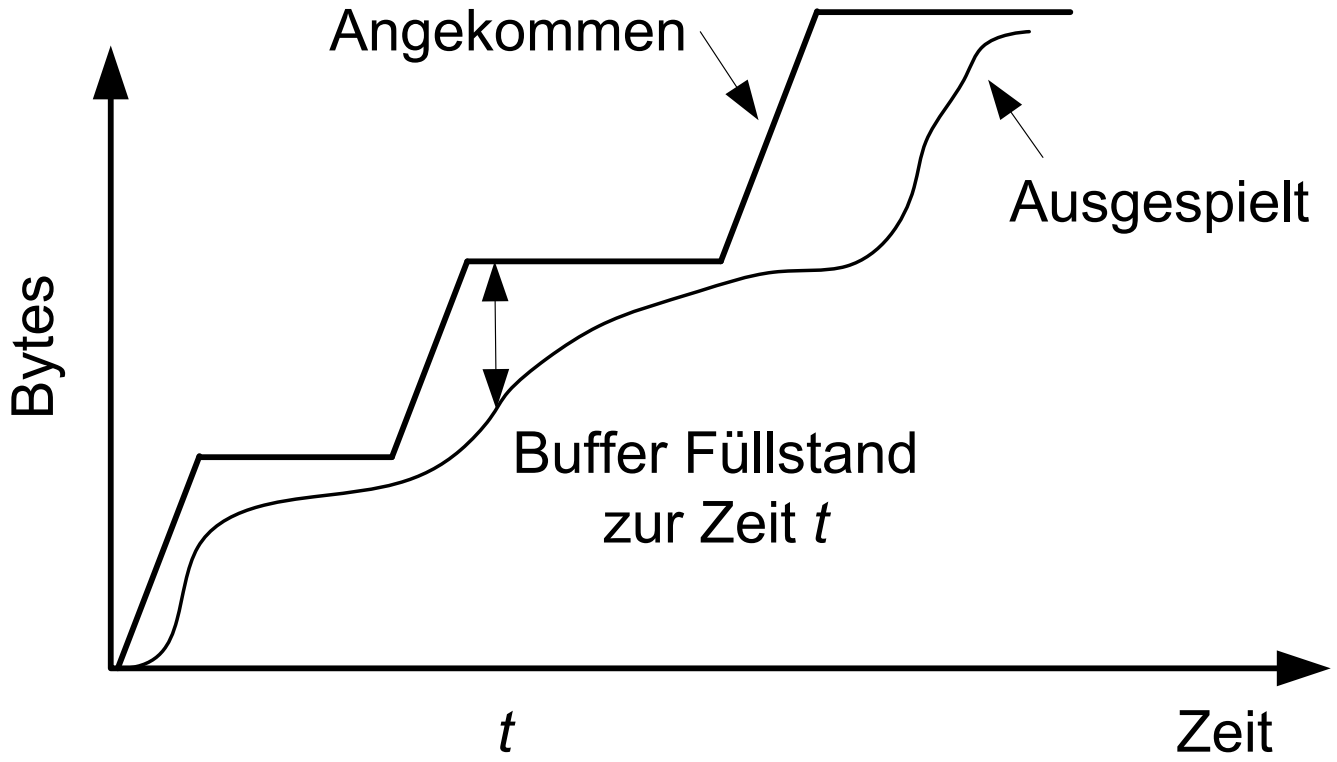
Puffer – Bulk Download



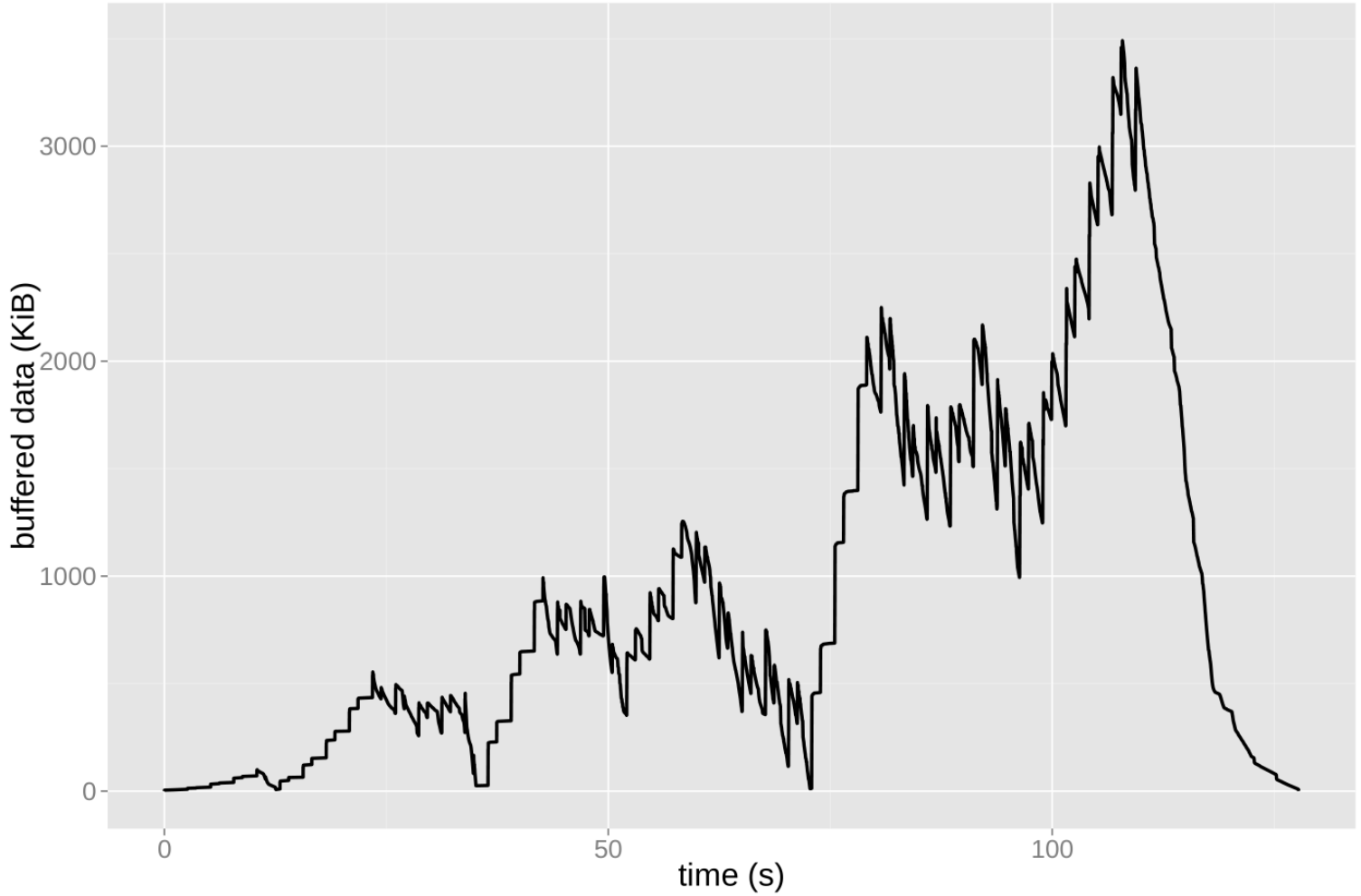
Puffer –VoD (progressiver DL)



Puffer –VoD (stufenweiser progressiver DL)



Puffer – Beispiel



Puffer – Probleme

- Pufferverzögerung zu groß
  - VoIP, Videokonferenz kaum möglich
- Pufferverzögerung zu klein
  - *Buffer underrun*
  - Netzwerk-Jitter ist zu hoch, Pakete werden zu stark verzögert und kommen nicht rechtzeitig an
  - Erreichte Bandbreite kleiner als Videobitrate
  - Paketverluste
- *Buffer overflow*
  - Wenn für eine gegebene Situation zu wenig Puffer-Speicher reserviert wurde

3. Streaming

- 3.1 Netzwerkeigenschaften
- 3.2 Streaming Prinzipien
- **3.3 Protokolle**
- 3.4 Voice over IP
- 3.5 Fehlertoleranz

Multimedia über das Internet

- Best Effort Netzwerk
  - keine QoS Garantien, Gleichbehandlung
  - Ende-zu-Ende Prinzip, keine „Intelligenz“ im Netz
  - Techniken auf der Anwendungsschicht, um Auswirkungen von Verzögerungen und Verlusten zu minimieren
  - Bei Problemen:
    - Noch mehr Bandbreite („Overprovisioning“), CDNs
    - „Schlaue“ Anwendungen (z.B. Abspielstrategien)
  - Alternative Ansätze
    - Ende-zu-Ende Reservierungen (IntServ)
    - Einführung von Dienstklassen (DiffServ)
    - Nicht mit den Internet-Grundprinzipien vereinbar

TCP/HTTP Streaming

- HTTP
  - Allgemeines Dateiübertragungsprotokoll (GET-Requests)
  - Protokoll zustandslos, Client kontrolliert Streaming
- Progressive Streaming
  - Videodaten werden in Dateien auf einem Web-Server abgelegt und per HTTP übertragen
  - Während der Übertragung wird Video bereits abgespielt
    → *Streaming*
  - Abspielen z.B. direkt im Browser/Anwendung oder über Plugins (Flash, HTML5 <video>, …)
  - CDNs für bessere, Multicast-ähnliche Lastverteilung

Adaptives TCP/HTTP Streaming

- Videos in verschiedenen Qualitätsstufen am Server abgelegt (z.B. 360p, 480p, 720p, 1080p)
- Videos segmentiert in gleichlange Stücke (z.B. 2-10s je)
- Client fordert je nach Abspielstrategie und aktuellen Netzwerkbedingungen die Segmente in der korrekten Qualitätsstufe an und spielt diese ab.
  - (Oder: unsegmentiert aber mit guten Einsprungpunkten (I-Frames und Metadaten) und HTTP Range Requests)
- Live Streaming möglich
- Verschiedene Standards für Dateiformate und Metadaten
  - Apples HTTP Live Streaming (IETF Draft)
  - MPEG DASH (verwendet z.B. von YouTube)
  - Abspielstrategien i.d.R. nicht standardisiert

RTP Streaming

- Spezialisiertes Streaming-Protokoll (im Gegensatz zu HTTP)
- Sammlung mehrerer zusammengehöriger Protokolle
  - RTP: Real-time Transport Protocol; Videotransport
  - RTSP: Real-time Streaming Protocol; Streaming-Kontrolle
  - RTCP: Real-time Control Protocol: Status-Informationen
- Multicastfähig über dedizierte Multicastadresse
- RTP ohne RTSP/RTCP auch häufig in Kombination mit anderen Protokollen verwendet, z.B.
  - SIP
  - WebRTC
  - XMPP mit Jingle

RTP (RFC 3550)

- UDP-Paketformat für die Übertragung von Audio und Video
- Streamkontrolle serverseitig (push), Client empfängt nur
- Reines Übertragungsformat, keine Steuerung
- RTP-Stream beinhaltet immer nur einen Medientyp
  - Für Video mindestens 2 RTP-Streams nötig (Audio+Video separat)
  - Müssen beim Client wieder synchronisiert werden
- Header

  - Payload-Typ: Codierung des Payloads. Kann während der Übertragung gewechselt werden (z.B. 0: PCM μ-law, 33: MPEG2)
  - Sequenznummer: Erkennung von Paketverlusten und Wiederherstellen der richtigen Paketreihenfolge
  - Zeitmarke: Erstellungszeitpunkt des ersten Samples im Paket

RTSP (RFC 2326)

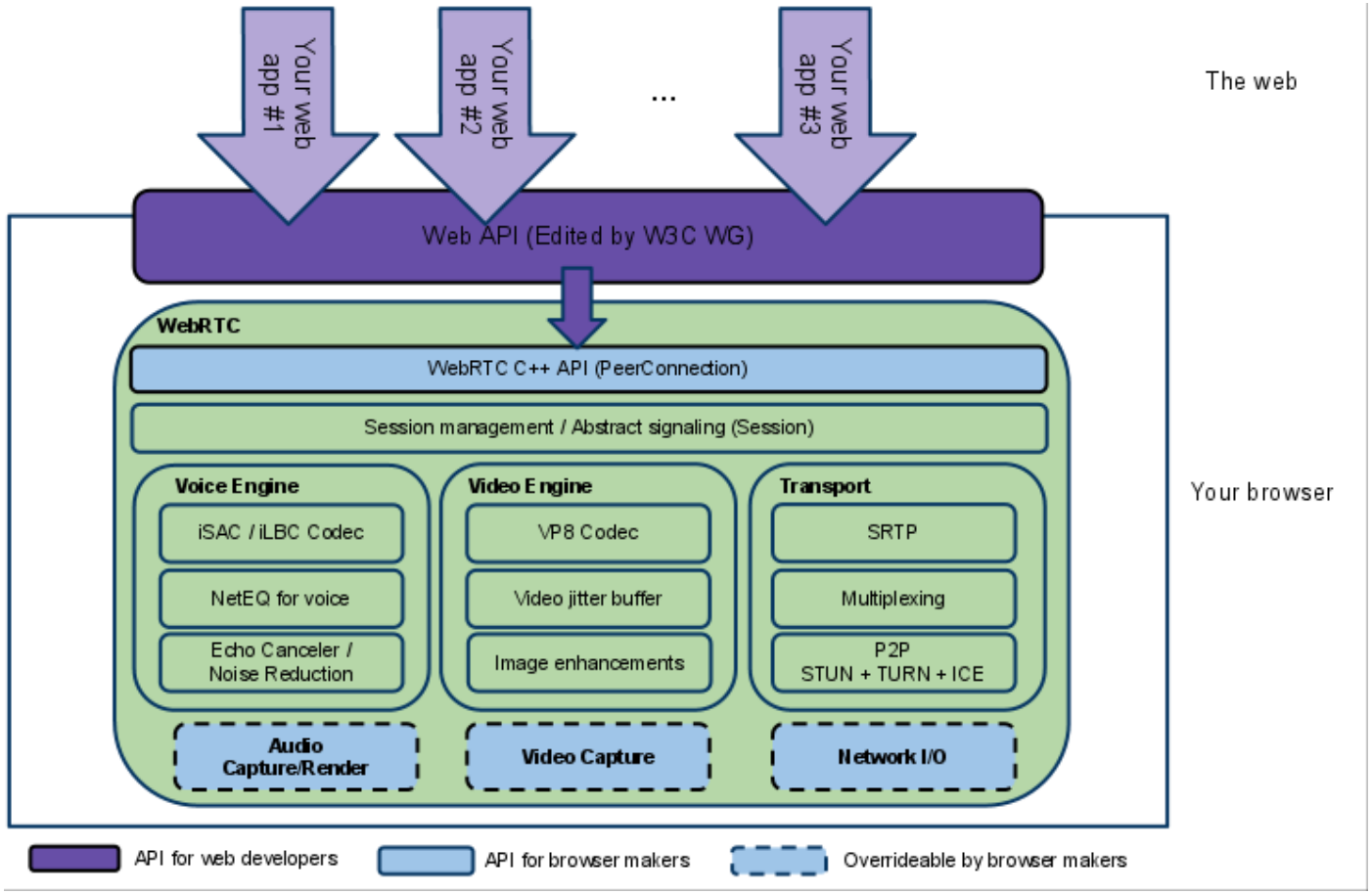
- Kontroll-/Signalisierungsprotokoll zur Steuerung mehrerer RTP-Streams
  - „Out-of-Band“-Signalisierung
  - SETUP, PLAY, PAUSE, OPTIONS, …
- TCP-basiert (Signale sollen immer ankommen)
- Verwendung bei Unicast
- Keine Übertragung der eigentlichen Mediendaten
- Keine Kontrolle über das Puffern der Daten im Player
  - Kontrolle liegt allein beim Server

RTCP (RFC 3550)

- Server in Kontrolle, braucht daher Informationen über Empfänger und Empfangsqualität
- RTCP: Sitzungsprotokoll zur Erhebung der
  - Informationen über Teilnehmer und deren Empfangsqualität
  - Receiver Report Verlustrate, letzte Sequenznummer, Jitter
  - Sender Report: SSRC, Zeitstempel, Zahl der gesendeten Pakete/ Bytes, genutzt zur Streamsynchronisation
  - Source Description: Teilnehmerinformationen
- RTP-Sender kann auf Informationen reagieren und Übertragung anpassen
- UDP, auch einfach bei Multicast einsetzbar
- Periodische Übertragungen zwischen allen Teilnehmern (max. 5% der RTP Bandbreite)

WebRTC (In Standardisierung IETF/W3C)

- Protokollarchitektur für Peer-to-Peer Multimedia-Kommunikation zwischen Browsern



3. Streaming

- 3.1 Netzwerkeigenschaften
- 3.2 Streaming Prinzipien
- 3.3 Protokolle
- **3.4 Voice over IP**
- 3.5 Fehlertoleranz

Voice over IP (VoIP)

Verschiedene Ansätze

- Proprietär
  - Skype verwendet ein eigenes proprietäres Protokoll
  - Teamspeak
- Standardbasiert aber nur provider-intern betrieben
  - z.B. SIP-Infrastruktur in einem „Walled Garden“, SIP-Deployment seitens der verschiedenen Telcos
- Standardbasiert und offen betrieben
  - Client-Server oder Peer-to-Peer Dienste
  - SIP, z.B. durch den Einsatz einer eigenen SIP-Infrastruktur, oder durch Internet-basierte Dienste wie z.B. SIPGATE
  - Mumble

VoIP Grundbegriffe

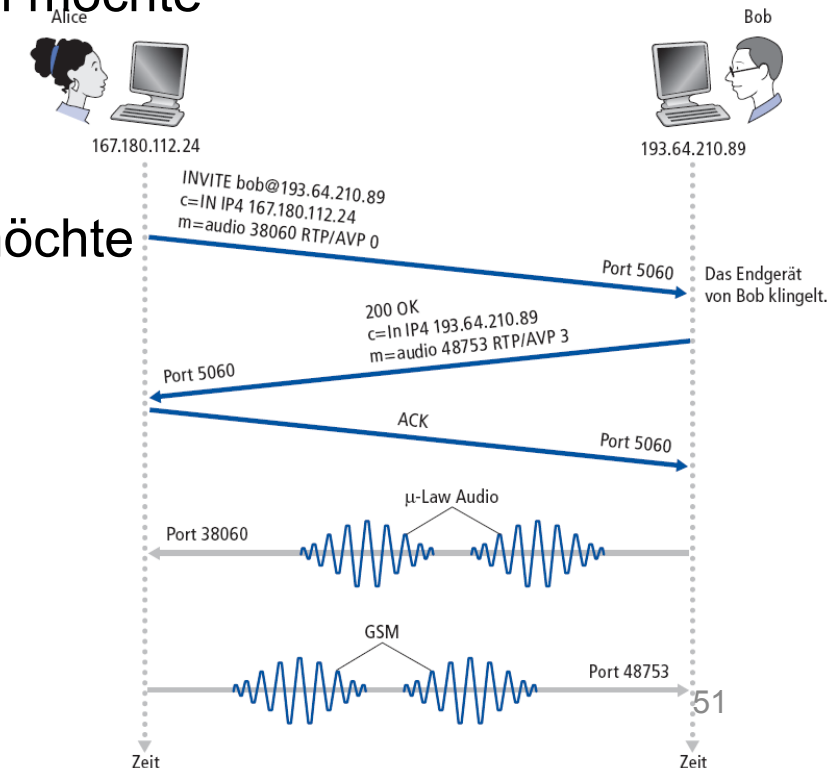
- VoIP-Server (z.B. SIP Proxy)
  - Call-Routing auf Applikationsebene
  - Auf- und Abbau von Gesprächen
  - Erweiterte Dienste: Konferenzen, Presence, Um- und Weiterleitungen
- Media-Server
  - Voice-Mediendaten, zB. Voice-Nachrichten, Abwesenheitsansagen
- IP Hardphone, Softphone
  - Physikalisches Telefon oder reine Softwareanwendung
  - Oft SIP-basiert
- Gateway
  - Verbindet VoIP-Telefonnetz mit herkömmlichem Telefonnetz (PSTN) oder mit anderen geschlossenen VoIP-Welten
  - Kann in die eigene Netzwerkstruktur eingebunden oder über externe Anbieter bezogen werden

Session Initiation Protocol (SIP, RFC 3261)

- Vision:
  - Alle Telefonanrufe und Videokonferenzen werden über das IP-Netzwerke geleitet
  - Menschen werden über ihren Namen oder ihre E-Mail-Adresse identifiziert und nicht über eine abstrakte Telefonnummer
  - Man kann den Kommunikationspartner erreichen, egal wo er sich aufhält und welches IP-fähige Endgerät er benutzt
- Verbindungsaufbau
  - Mechanismen, um den Kommunikationspartner darüber zu informieren, dass man eine Verbindung herstellen möchte
  - Aushandeln der zu verwendenden Medientypen und Codecs
  - Mechanismen für das Beenden einer Verbindung
- Bestimmen der aktuellen IP-Adresse des Kommunikationspartners
  - Abbildung eines Namens auf die dazugehörige IP-Adresse
- Anrufsteuerung
  - Neue Medienströme zu einer Verbindung hinzufügen
  - Codec während einer Verbindung ändern
  - Neue Teilnehmer hinzufügen

Herstellen einer Verbindung

- Zu einer bereits bekannten IP-Adresse über UDP oder TCP
- *Invite*-Nachricht von Alice
  - IP-Adresse + Portnummer
  - Codierung, die Alice empfangen möchte
- *OK*-Nachricht von Bob
  - IP-Adresse + Portnummer
  - Codierung, die er empfangen möchte



Herstellen einer Verbindung

- Aushandeln der Codierung
  - Falls Bob gewünschte Codierung nicht bereitstellen kann
  - Antwort mit *606 not acceptable*
  - Angabe einer Liste von unterstützten Codierungen
  - Erneutes *Invite* durch Alice mit unterstütztem Format
- Ablehnen eines Rufes
  - Bob kann einen Ruf ablehnen und Response Code angeben (z.B. 402 payment required, 403 forbidden, 410 gone, 486 busy)
- Anschließend
  - Session Description Protocol (SDP, RFC 2327) zur Übermittlung der Stream-Details (Codecs, A/V Kanäle, Konfigurationen, …)
  - Übertragung der eigentlichen Mediendaten mit z.B. RTP

Abbildung des SIP-Namens auf IP-Adresse

- Anrufer kennt lediglich den SIP-Namen des Gesprächspartners
- Er muss die IP-Adresse herausfinden
  - Benutzer sind mobil
  - DHCP wird eingesetzt
  - Benutzer haben mehrere Geräte mit IP-Adressen (PC, PDA, usw.)
- Das Ergebnis der Anfrage kann abhängen von:
  - Tageszeit (am Arbeitsplatz, zu Hause)
  - Anrufer
  - Status des angerufenen Teilnehmers (z.B. Umleitung auf Voicebox, wenn man gerade mit einer anderen Person spricht)
- **Von SIP angebotene Dienste:**
  - SIP-Registrar
  - SIP-Proxy-Server
  - Funktionalität ähnlich zu DNS

SIP-Registrar

Wenn Bob seinen SIP-Client startet, dann schickt dieser eine SIP REGISTER Nachricht an seinen Registrar-Server

Beispiel für eine solche Nachricht:

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

SIP-Proxy

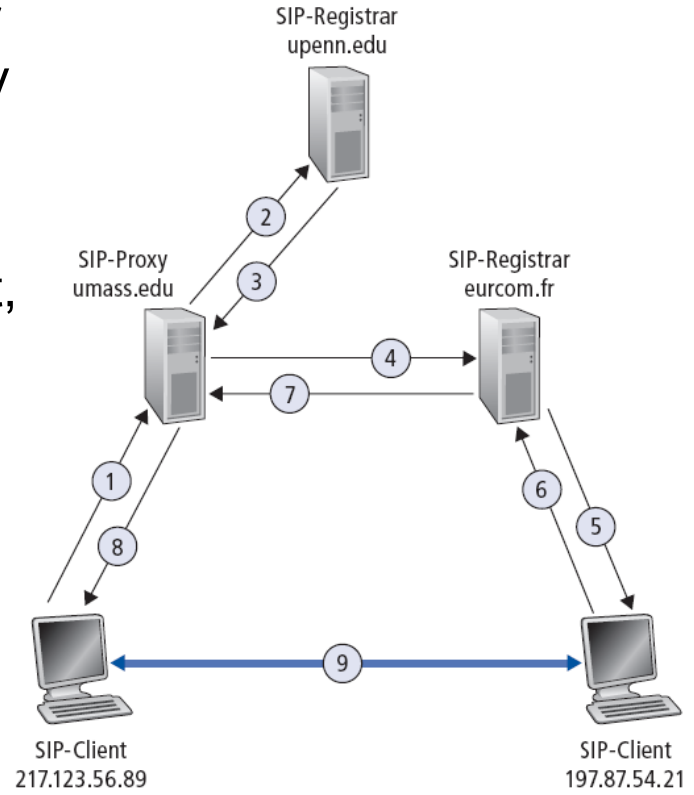
- Alice sendet ihre *Invite*-Nachricht an ihren Proxy-Server
  - *Invite* beinhaltet die Information: sip:bob@domain.com
- Der Proxy Server ist dafür verantwortlich, die SIP Nachricht zum angerufenen Teilnehmer weiterzuleiten
- Der angerufene Teilnehmer sendet seine Antwort auf dem gleichen Weg zurück
- Der Proxy von Alice liefert die Antwort an Alice aus
  - In der Antwort ist die IP-Adresse von Bob enthalten

SIP-Proxy - Beispiel

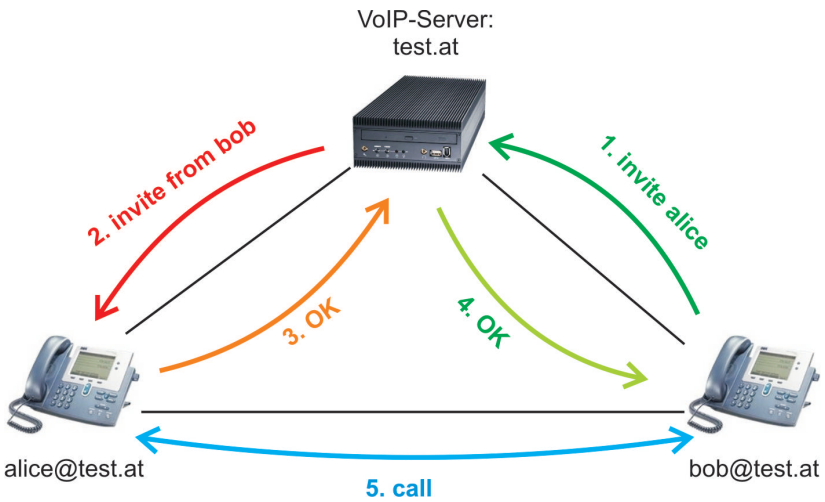
1. jim@umass.edu sendet Invite-Nachricht für keith@upenn.edu an den `umass`-SIP-Proxy
2. Proxy leitet dies an den `upenn`-Registrar weiter
3. `upenn`-Registrar gibt eine Redirect-Antwort, die besagt, dass man das Ziel nun unter `keith@eurecom.fr` erreichen kann
4. Proxy schickt ein INVITE an den `eurecom`-Registrar
5. Der `eurcom`-Registrar leitet das Invite an `197.87.54.21` weiter

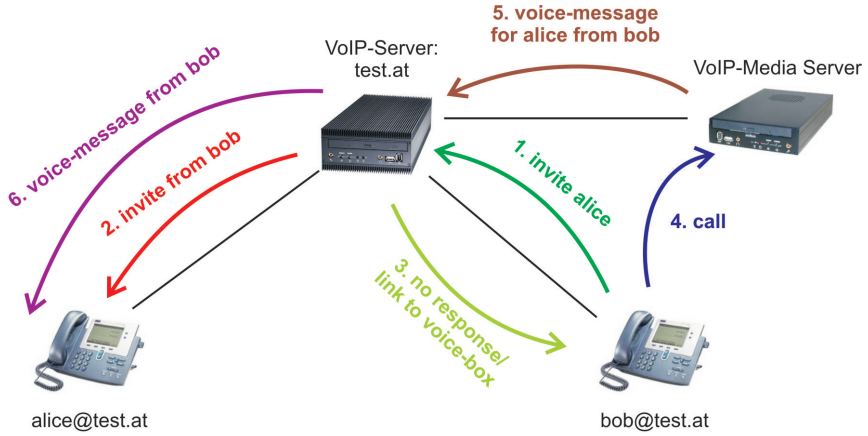
6.-8. SIP-Antworten werden zum Anrufer geschickt

9. Aufbau einer direkten Verbindung

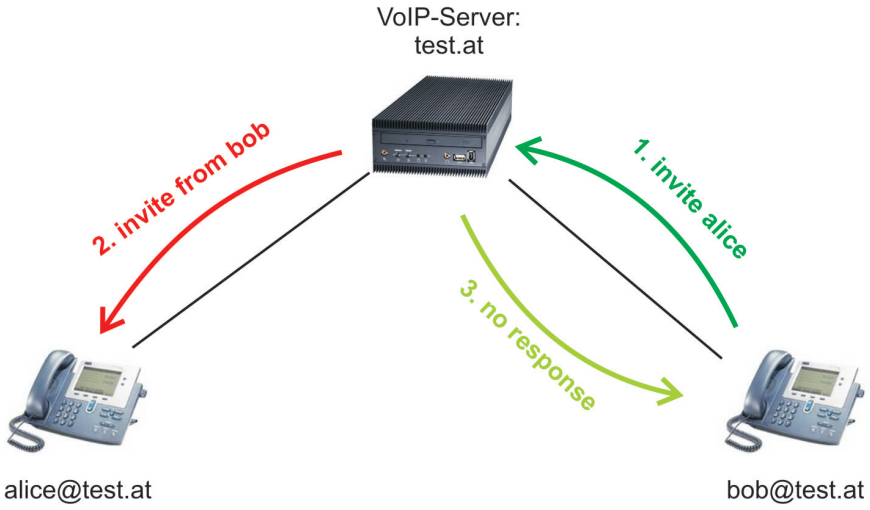


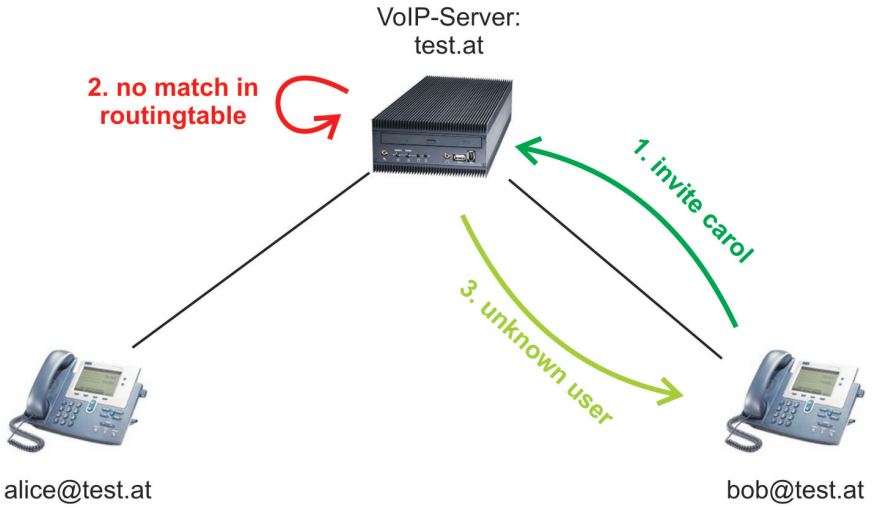
SIP Szenarien





- 6. after next registration of alice





Client-Server VoIP

- XMPP+Jingle
  - XMPP RFC 6120, 6121, 6122, Jingle als Draft
  - Instant Messaging Protokoll
  - Absicherung mit SSL und OTR
  - Server-Federation
  - In Abwandlung verwendet von Google, Facebook, WhatsApp, …
  - Jingle als A/V Kommunikationserweiterung (Transport über RTP)
- Mumble
  - Open Source VoIP, nicht standardisiert
  - Spezialisiert auf Low Latency (z.B. durch Opus)

3. Streaming

- 3.1 Netzwerkeigenschaften
- 3.2 Streaming Prinzipien
- 3.3 Protokolle
- 3.4 Voice over IP
- **3.5 Fehlertoleranz**

Netzwerkprobleme

- IP-Netze sind in ihrer Grundform „Best-Effort“-Netze, und daher ist Netzwerküberlastung möglich
  - Am Server → zu viele Clients
  - Im Netzwerk (Links, Router)
    → **Die Ende-zu-Ende Bandbreite ist nicht garantiert!**

- Metriken zur Bestimmung der Netzwerkgüte:
  - RFC 4148: IPPM (IP Performance Metrics)
    - Paketverlust
    - Verzögerung
    - ...

Strategien gegen Netzwerkprobleme

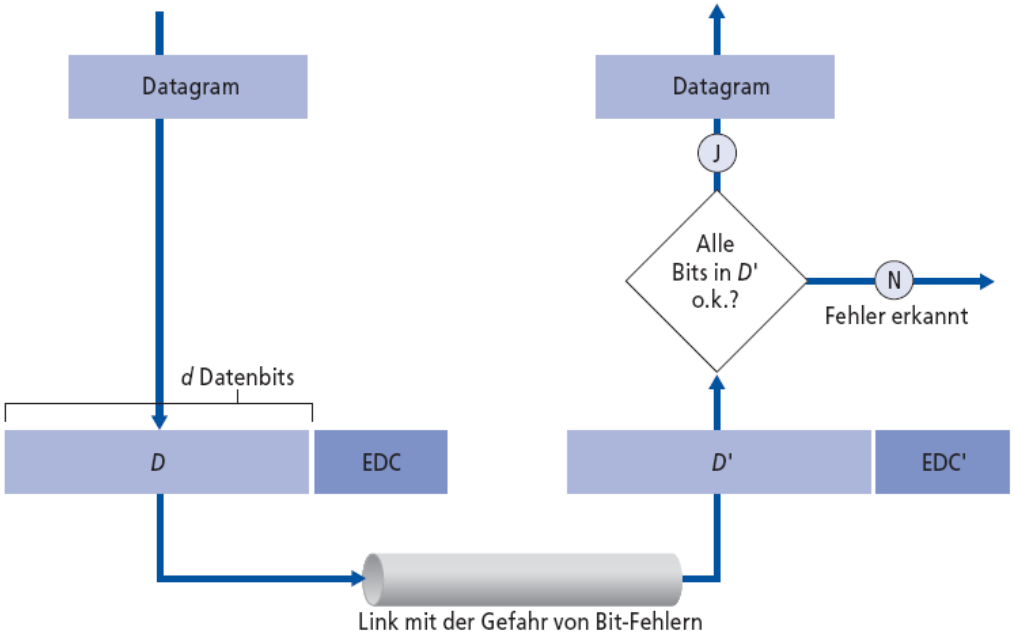
- Fehlererkennung (Error Detection)
- Reaktion (Adaption), Auswirkungen vermindern
- Fehlerkorrektur durch Redundanz
- Verstecken des Fehlers beim Empfänger (Concealment)
- Robuste Kodierung (Resilience)

Netzwerkprobleme auf der Linkebene

- Sicherungsschicht (Layer 2)
- Probleme auf einem einzelnen Link
- Anpassung der Übertragungsparameter
  - Signalstärke
  - Kodierung
- Anwendung von Fehlererkennung / Korrektur
  - Prüfsummen
  - CRC
  - Retransmission

Fehlererkennung

- Die Fehlererkennung ist nie 100% zuverlässig!
  - Error Detection and Correction Bits (EDC)
    - Nützliche Redundanz
  - Bestimmte Bitfehlermuster können übersehen werden
  - Mehr EDC-Bits führen zu besseren Erkennungsraten



Fehlererkennung – Linkebene

- Ziel: Erkennen von Fehlern in übertragenen Segmenten auf der Transportschicht
- Parity Bit (XOR)
  - Statt n Bits sende n+1 Bits
  - Letztes Bit…Parity Bit (PB)
  - Summe über alle Bits gerade → PB=0
  - Summe über alle Bits ungerade → PB=1

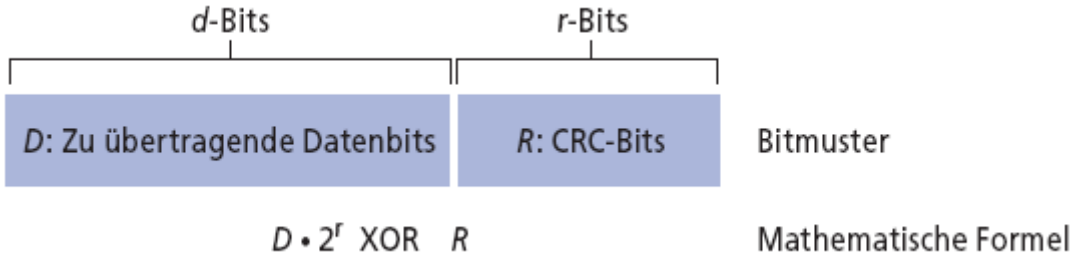
Fehlererkennung – Linkebene

- Cyclic Redundancy Check (CRC)
  - Statt k Bits sende n>k Bits
  - Letzten (n-k) Bits → CRC
  - Berechnung:
    - Daten (Bits) werden als Polynom betrachtet
    - Polynom wird durch ein bekanntes/gegebenes kleineres Polynom dividiert
    - CRC ist der Rest dieser Division

Fehlererkennung – Linkebene
Cyclic Redundancy Check (CRC)

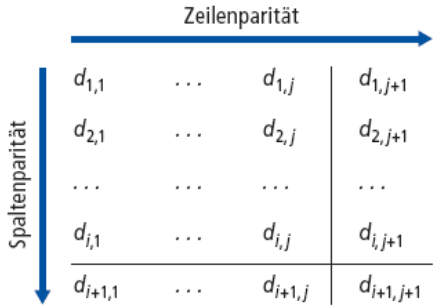
- Betrachte die Datenbits (D) als eine binäre Zahl
- Wähle ein Bitmuster der Länge r+1 (Generator, G)
- Ziel: Wähle r CRC-Bits (R) so, dass gilt:
  - <D,R> ist Modulo 2 durch G ohne Rest teilbar
  - Empfänger kennt G und teilt das empfangene <D',R'> durch G. Wenn es einen Rest gibt: Fehler erkannt!

  → Kann alle Burst-Fehler erkennen, die kürzer als r+1 Bit sind
- In der Praxis weit verbreitet (IEEE 802.11 WLAN, ATM)

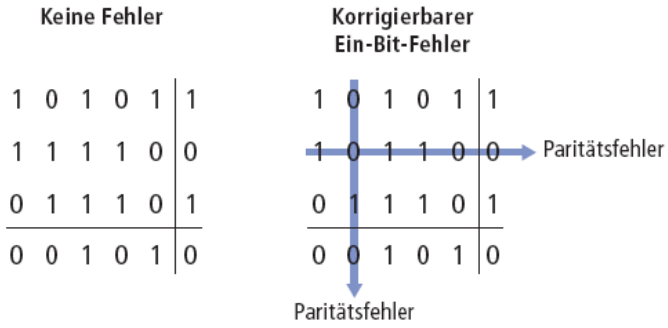


Forward Error Correction – Linkebene

- FEC (Forward Error Correction)
  - Statt k Bytes sende n>k Bytes
  - Anzahl fehlerhafter Bytes < obere Schranke
  - ➢ Fehlerhafte Bytes können beim Empfänger korrigiert werden!

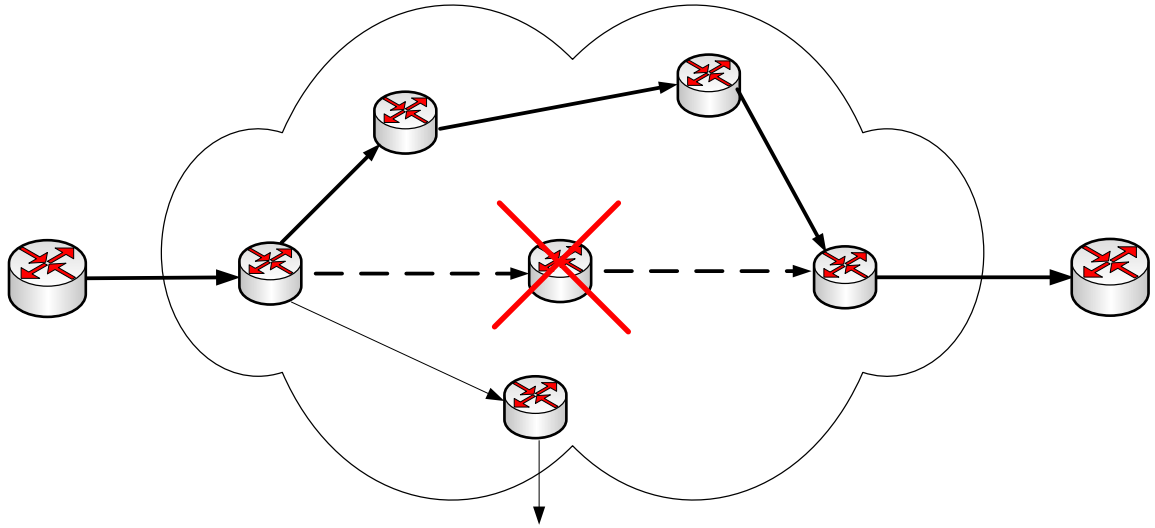


- Zweidimensionale Parität:
  - Erkennt und korrigiert Ein-Bit-Fehler



Netzwerkprobleme auf der Netzwerkebene

- Netzwerkschicht (Layer 3)
- Reaktion auf Knoten / Pfadausfälle
- IP überträgt verlorene Pakete nicht neu
- Anpassung des Routings entsprechend dem Algorithmus



Fehlererkennung – Netzwerkebene

- Routing → Beispiel OSPF (Link State):
  - Router kennen ihre Nachbarn und sie tauschen mit ihnen periodisch Nachrichten aus (Hello Pakete)
  - Falls ein Nachbar-Router für einige Zeit nicht antwortet, wird der Link zu ihm als ausgefallen betrachtet, gefolgt von diesen Schritten:
    - Die entsprechende Link-State Information wird im Netz geflutet
    - Nach erhalt der neuen Link-State-Information(en) berechnen alle Router innerhalb des Netzes ihre Routing-Tabellen neu, wodurch sich netzweit ein neuer konsistenter Zustand ergibt
  - Wenn der ausgefallene Link/Router wiederhergestellt ist, werden die Schritte (1) und (2) erneut durchgeführt

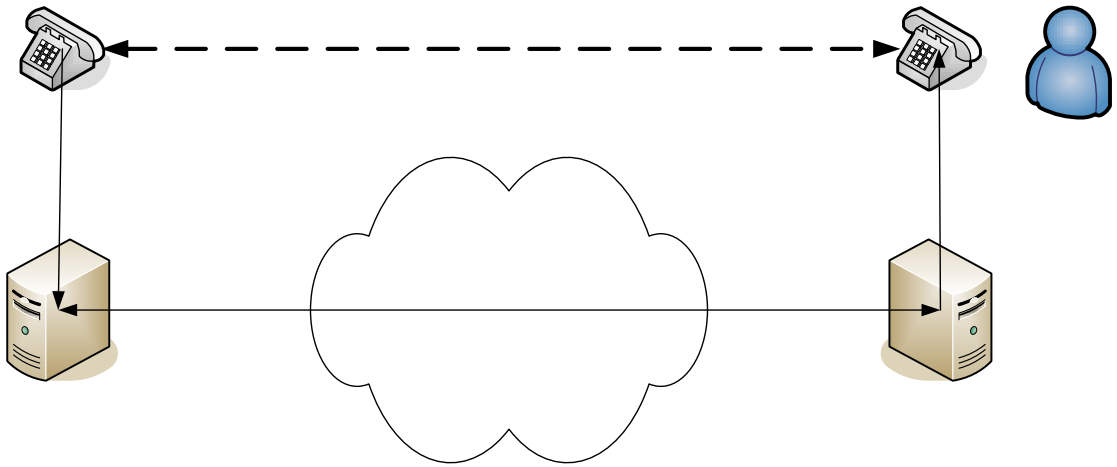
Fehlererkennung - Transportschicht

- Sender:
  - Betrachte das Segment als eine Folge von 16-Bit-Integerwerten
  - Prüfsumme: Addition (im 1er-Komplement) der Werte
  - Sender schreibt das Ergebnis in das TCP/UDP-Prüfsummenfeld

- Empfänger:
  - Berechne die Prüfsumme
  - Passt diese zum Wert im Prüfsummenfeld:
    - Nein – Fehler erkannt
    - Ja – kein Fehler erkannt. Aber es könnten dennoch Fehler vorliegen!

Netzwerkprobleme aus Sicht der Applikationsebene

- Für Layer 7 ist das Netzwerk eine „Black Box“
- Intelligente Applikationen passen die Übertragungsparameter der gegebenen Packet-Loss-Rate bzw. Ende-zu-Ende Bandbreite an



Fehlererkennung – Applikationsebene

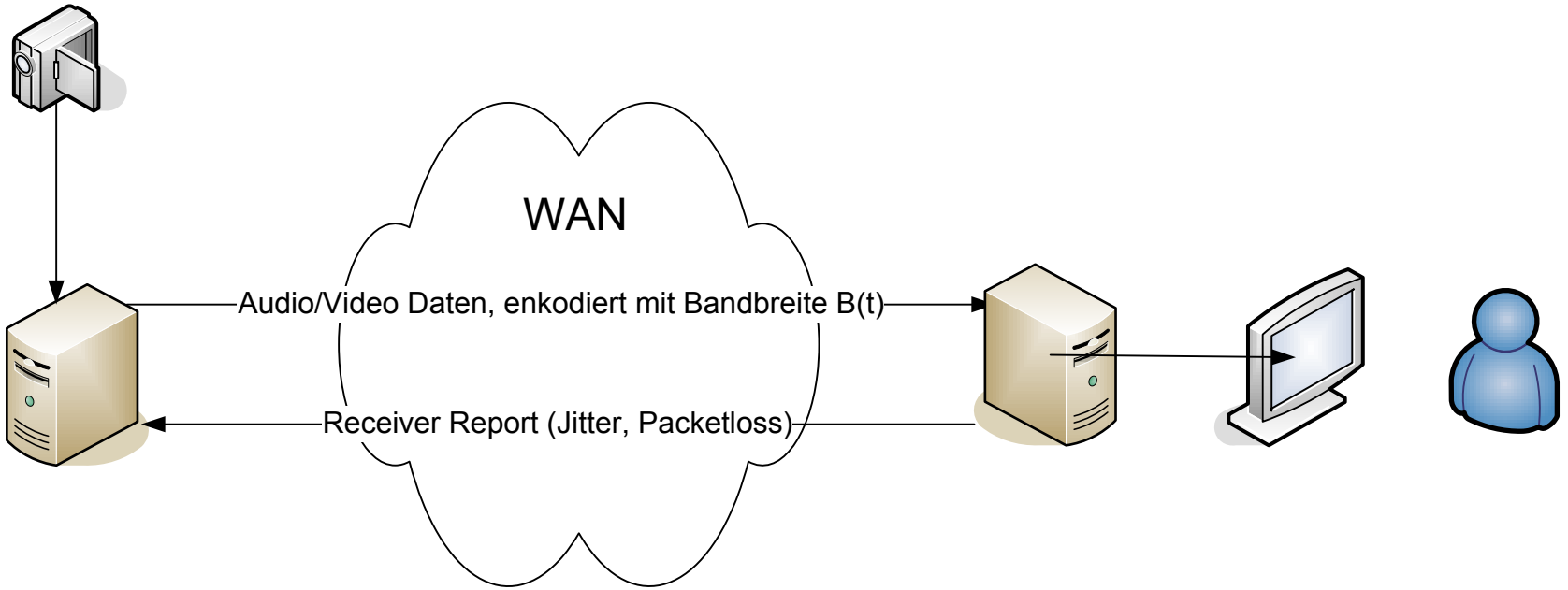
- Erkennung von Paketverlusten (Packet Loss Detection)
  - Continuity Counter (z.B. RTP)
  - Jedes Paket erhält eine neue Nummer
  - Client sieht diese Nummern
  - Falls eine Lücke auftritt weiß der Client → Paket ist ausgefallen/ verspätet

Streaming Control Loop (1/2)

- Einfache Control-Loop (“Feedback”) zwischen Sender und Player

- Player sieht die ankommenden Pakete
  - Schätzung End-to-End Bitrate
  - Verspätete/Verlorene Pakete
- Player schickt an den Server einen Bericht (Report)
  - siehe RTCP
- Falls End-to-End Bandbreite zu klein
  - Server adaptiert die Bitrate

Streaming Control Loop (2/2)



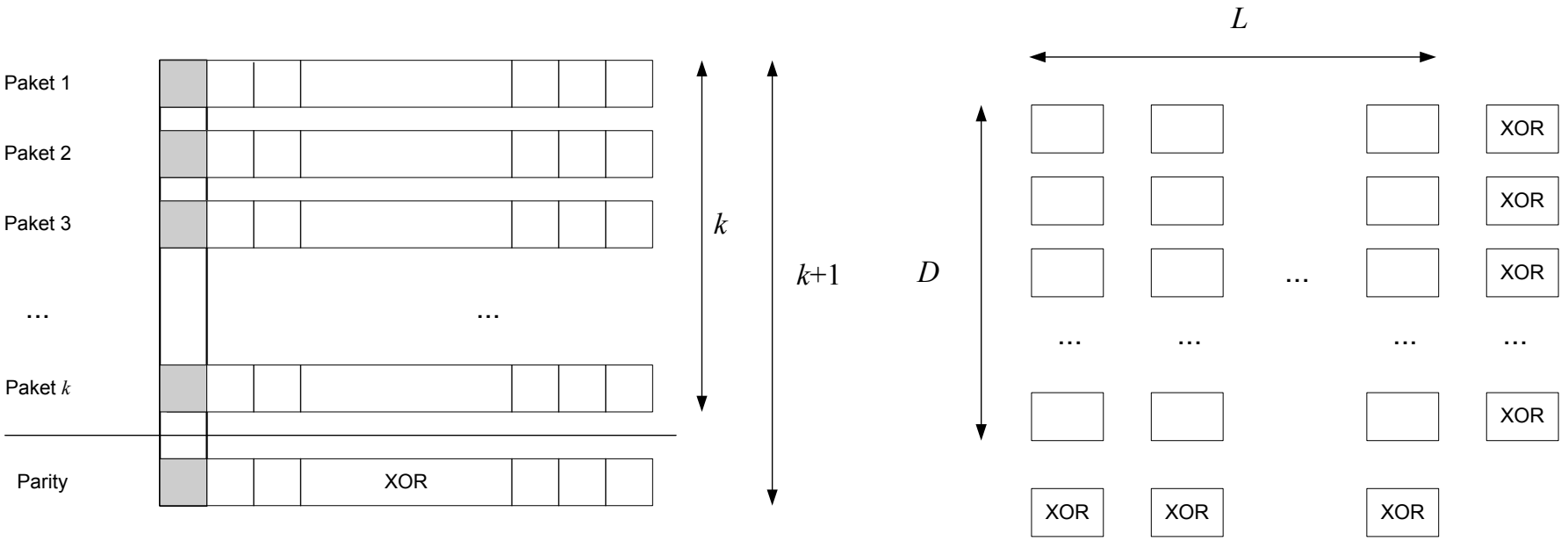
Adaptation, Reaktion

- Applikationen reagieren
  - Pakete werden erneut geschickt
  - Pakete werden aufwendiger codiert (FEC)
  - Es werden weniger Daten geschickt
- Netzwerkmanagement-Tools reagieren
  - Logischer Pfad wird umgeleitet (z.B. mittels MPLS)
- Link reagiert
  - Frames werden erneut geschickt
  - Frames werden aufwendiger codiert (FEC)
  - Modulationsanpassung (QAM), Anpassung der Sendestärke
  - MIMO Beamforming
  - Usw.

XOR – Applikationsebene

- Server schickt statt k **Paketen** $k+1$ Pakete
- Alle Pakete gleich lang, eindeutig identifizierbar
- FEC = Byteweises XOR der Daten
- Falls 1 Paket ausfällt kann es **rekonstruiert** werden
- Vorteil XOR: einfach, **geringer Rechenaufwand**

XOR – Applikationsebene



Error Concealment

- Multimedia Daten sind meistens hoch redundant

- Error Concealment
  - Verstecke Fehler bzw. verlorene Daten so gut es geht
  - Betrachter sollte die Auswirkung kaum wahrnehmen
  - Audio
    - Verlorene Samples werden wiederholt (G.711) oder interpoliert
  - Video
    - Ausgefallener Slice: alter Bildbereich wird einfach weiter angezeigt
    - Interpolation von DC-Komponenten

Robuste Kodierung

- Anforderungen:
- Wirkung von Übertragungsfehlern soll minimiert werden (nur lokale Auswirkungen)
- Geringer Einfluss auf die Kodiereffizienz
- Fehler:
  - Bemerken
  - Lokalisieren
  - Dekodierprozess soll schnell fortgesetzt werden

Robuste Kodierung

- MPEG-4 AVC
  - Paketorientiert
  - Network Abstraction Layer (NAL) Units
  - Eine Slice pro NAL Unit
- Eine NAL Unit enthält **entweder**
  - **Videodaten** (Slice)
  - Parameter, die die Kodierung beschreiben (**Parameter Set**)
  - Supplemental Enhancement Information (**SEI**): z.B. Timing
- Bereits optimiert für Übertragung per **RTP** über **paketorientierte** Netzwerke

AVC

- Picture Segmentation
  - Bilder sind in Slices unterteilt
  - Der Encoder passt die Slice-Größe der vorhergesehenen Pfad-MTU an
  - Dann passt jedes Slice in ein Paket und Fragmentierung wird vermieden (→ günstiger bei Paketverlusten)
- Arbitrary Slice Ordering
  - Slices können unabhängig voneinander dekodiert werden
  - Müssen nicht mehr im Puffer auf verspätete Slices warten
  - Echtzeitdekoder kann Slices, die zu früh angekommen sind, vorzeitig dekodieren
  - ➢ Verbesserung für Echtzeitkommunikation!

AVC – Redundant Slices

- Zu jedem Primary Slice (PS) kann man einen Redundant Slice (RS) kodieren
- RS wird mit wesentlich schlechterer Qualität kodiert
- PS und RS werden in unterschiedlichen Paketen transportiert
- Geht ein PS verloren, kann der Fehler durch das RS teilweise kompensiert werden!