



Systeme II

8. Multimedia

Christian Schindelhauer

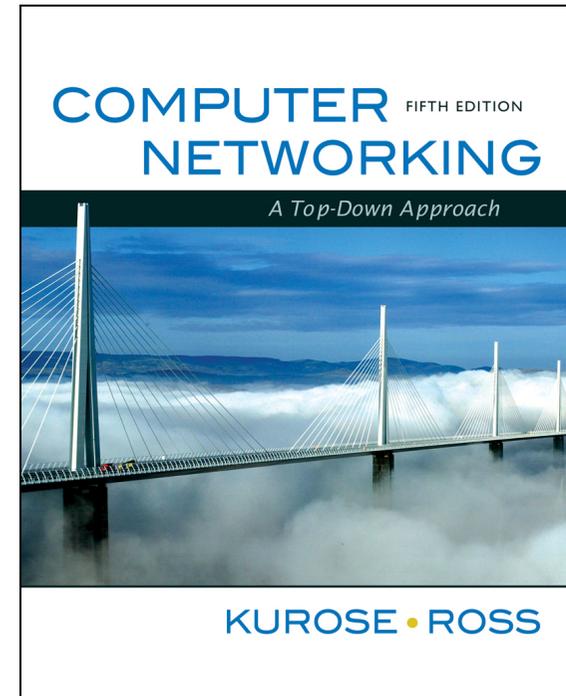
Technische Fakultät

Rechnernetze und Telematik

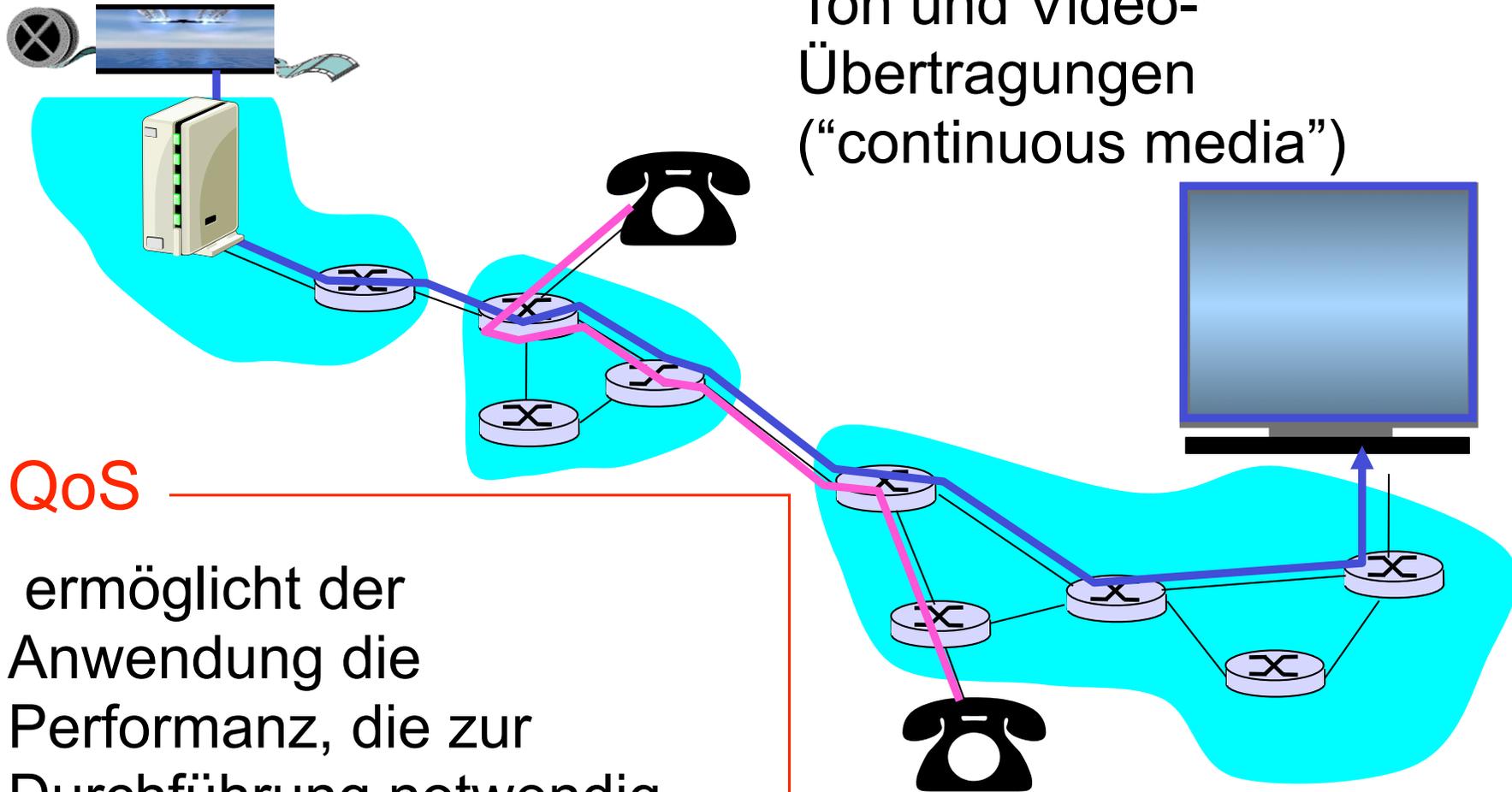
Albert-Ludwigs-Universität Freiburg

Version 08.07.2013

- Folien und Inhalte aus
 - Computer Networking: A Top Down Approach 5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.
 - Copyright liegt bei den Autoren Kurose und Ross



Multimedia Anwendungen Ton und Video- Übertragungen ("continuous media")



QoS

ermöglicht der
Anwendung die
Performanz, die zur
Durchführung notwendig

- Allgemeines
 - Klassifikation von Multimedia-Anwendungen
 - Erkennen der Bedürfnisse von Multimedia-Anwendungen
 - Best-Effort so gut wie möglich nutzen
- Protokolle und Architekturen
 - Protokolle für Best-Effort
 - Mechanismen für QoS
 - Architekturen für QoS

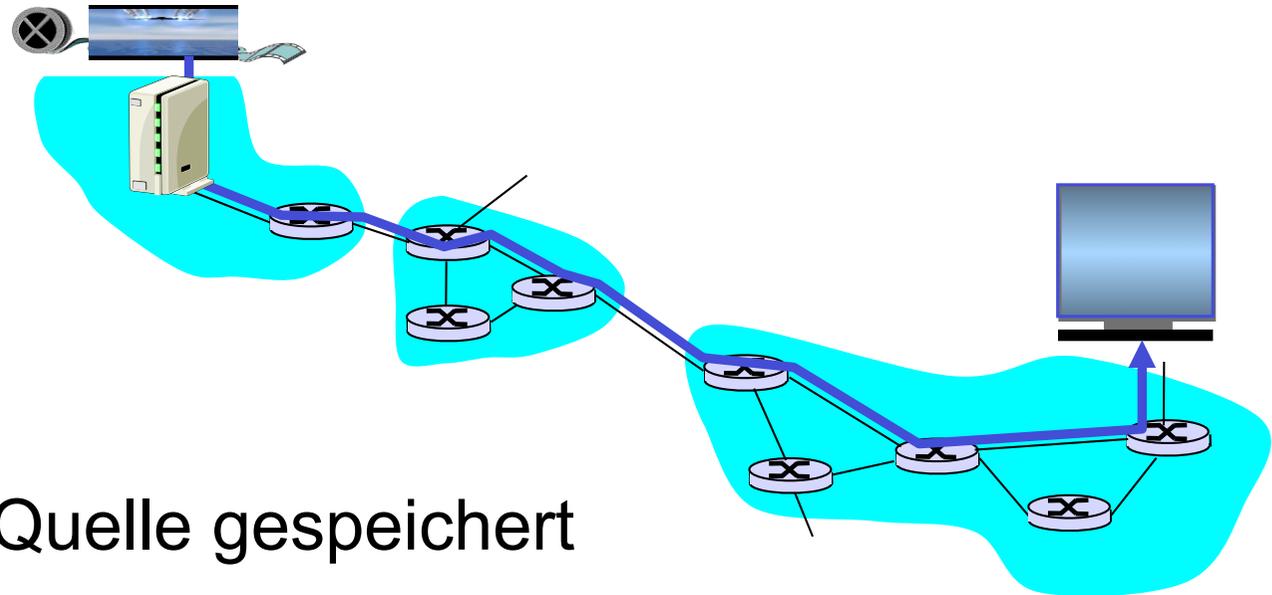
■ Klassen

- 1) Gespeicherte Ströme (stored streaming)
- 2) Live Streams
- 3) Interaktiv und Realtime

■ Typische Eigenschaften

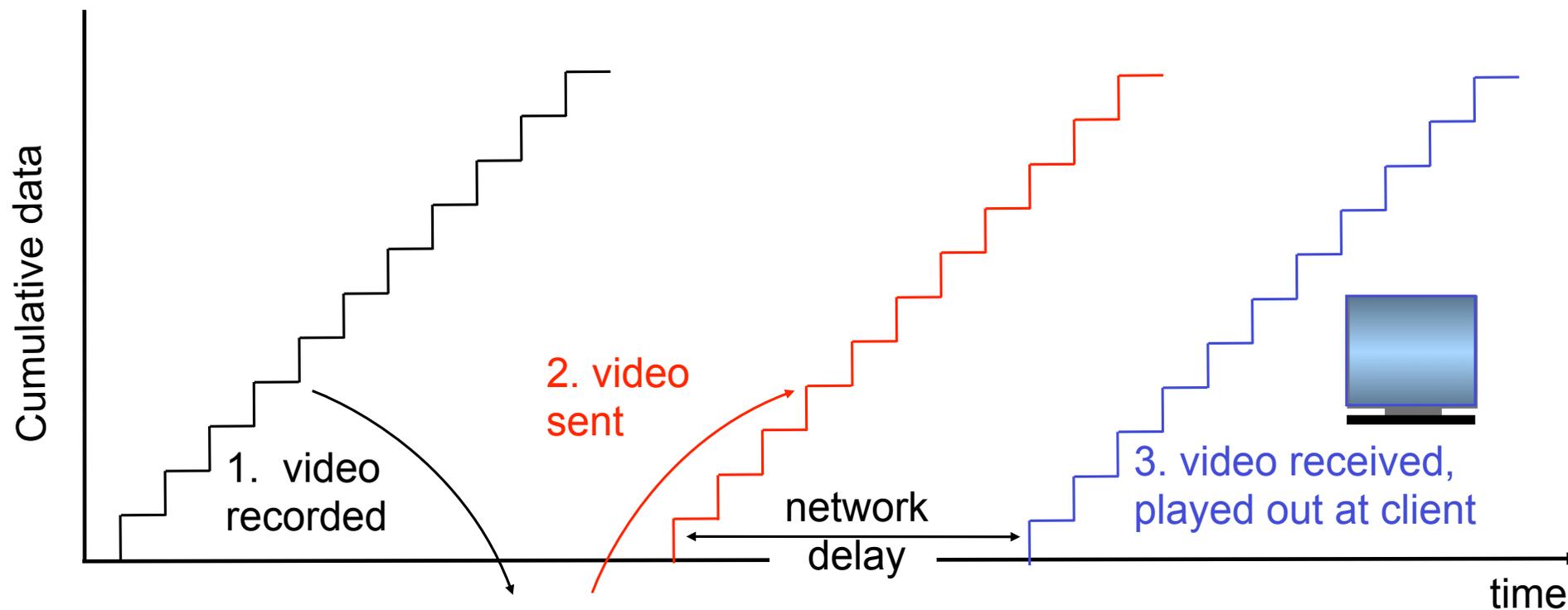
- Verzögerungs-Empfindlichkeit (delay sensitive)

- End-to-End Delay
- Delay Jitter
 - Jitter ist die Veränderung des Paket-Delays innerhalb des Paketstroms
- Toleranz von Paketverlusten:
 - seltene Verluste verursachen kleine Aussetzung
 - Gegenkonzept von Datenübertragung, die nicht Verluste toleriert, aber Delays



- Stored streaming:
 - Medium bei der Quelle gespeichert
 - Übertragung zum Client
- Streaming
 - Client spielt ab, bevor alle Daten angekommen sind

Gespeicherte Ströme (stored streaming)

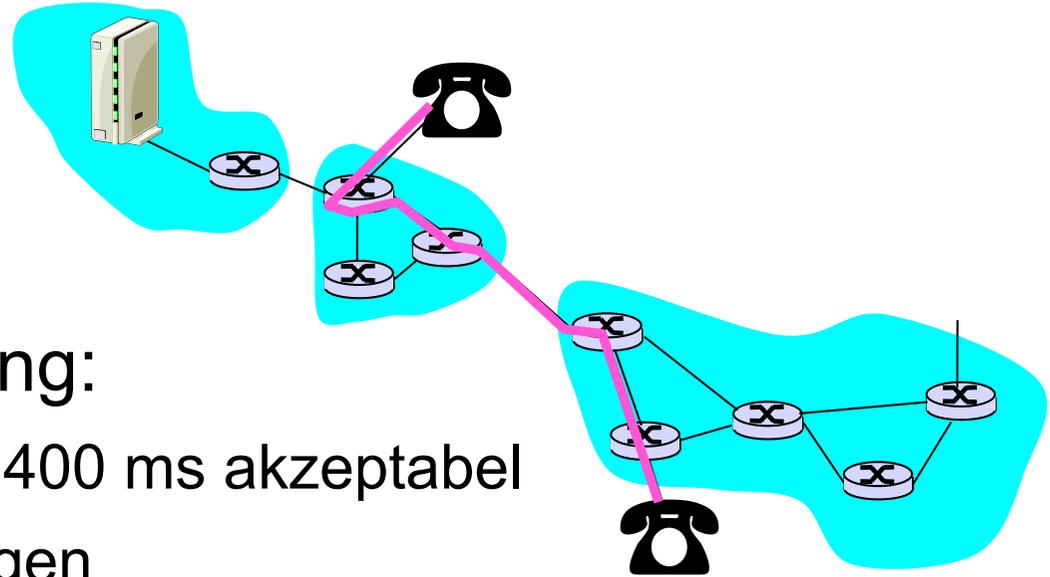


■ Streaming

- Client spielt ab, bevor alle Daten angekommen sind

- Beispiele:
 - Internet Radio
 - Sport Ereignis (Fußball..)
- Streaming
 - Wiedergabe-Puffer
 - Playback kann einige Zehntel Sekunden nach der Übertragung stattfinden
 - Dennoch Zeit-Constraints
- Interaktion
 - Vorspulen, Pause, und Zurückspulen möglich

- Anwendungen:
 - IP-Telefonie
 - Video-Konferenz
- Verzögerungsanforderung:
 - Audio: < 150 ms ist gut, < 400 ms akzeptabel
 - inklusive aller Verzögerungen
 - Anwendungsschicht und Netzwerk-Delay
- Größere Delays werden als Einschränkung empfunden
 - Initialisierung der Sitzung



- TCP/UDP/IP: “best-effort”
 - keine Garantien für Delay und Verlustraten
- Wie kann man QoS und Performanz erreichen?
 - Durch Methoden in der Anwendungsschicht
 - schwächen negative Einflüsse auf Delay und Verlust ab

- **Integration einer Service-Philosophie**
 - Fundamentale Veränderungen im Internet zur Reservierung von Bandweiten
 - benötigt neue, komplexe Software in Rechnern und Routern
- **Laissez-faire**
 - keine größeren Veränderungen
 - Mehr Bandweite soweit nötig
 - Verbreitung der Inhalte durch Multicast in der Anwendungsschicht
- **Differenzierte Service-Strategie**
 - kleine Veränderungen im Internet
 - Unterscheidung in erste und zweite Klasse-Pakete durch Priorisierung

- Analog-Sample-Rate
 - Telefon: 8,000 samples/sec
 - CD: 44,100 samples/sec
- Sample-Größe diskretisiert
 - z.B. $2^8=256$ mögliche Werte
- Werte durch Bits dargestellt
 - 8 bits for 256 values
- Beispiel: 8,000 samples/sec, 256 Werte ergibt 64,000 bps
 - Receiver wandelt Bits zurück zum Analogsignal um
 - mit gewissen Qualitätseinbußen
- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps
- Internet Telephonie: 5.3 kbps oder mehr

- Video: Bildsequenzen mit konstanter Rate
 - z.B. 24 Bilder/sec
- Digitales Bild: Pixelfeld
 - Jedes Pixel wird durch Bits dargestellt
- Redundanzen
 - im Raum (innerhalb des Bilds, z.B. eintönige Flächen)
 - in der Zeit (von einem Bild zum nächsten, z.B. Standbilder)
- Beispiele:
 - MPEG1 (CD-ROM) 1.5 Mbps
 - MPEG2 (DVD) 3-6 Mbps
 - MPEG4 (Internet, < 1 Mbps)
- Forschung:
 - geschichtetes (skalierendes) Video
 - Schichten passen sich an die Bandweite an

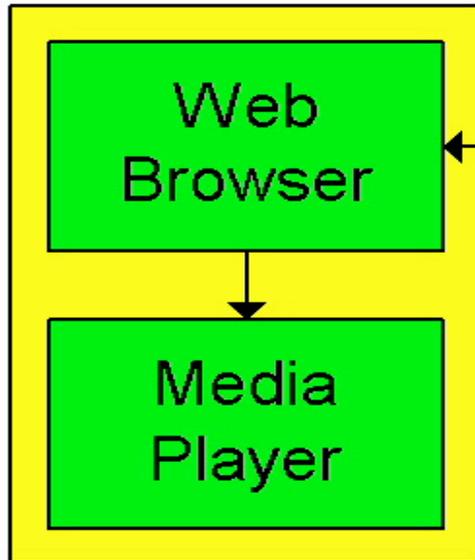
- Anwendungsschicht für Best-Effort-Service
 - Client-Puffer
 - UDP oder TCP
 - Verschiedene Kodierungsmethoden für Multimedia

Media Player

- jitter-Entfernung
- Dekomprimierung
- Fehlerbehandlung
- GUI

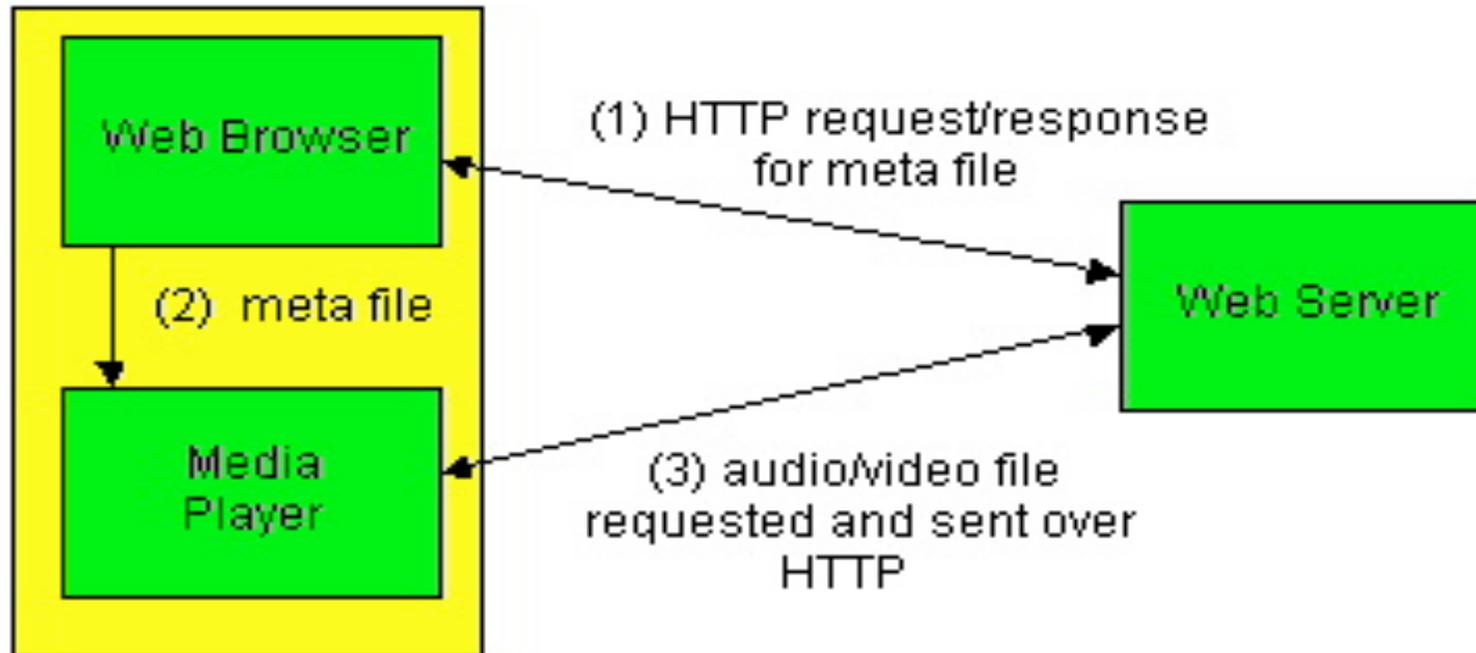
Internet Multimedia

Der einfachste Ansatz

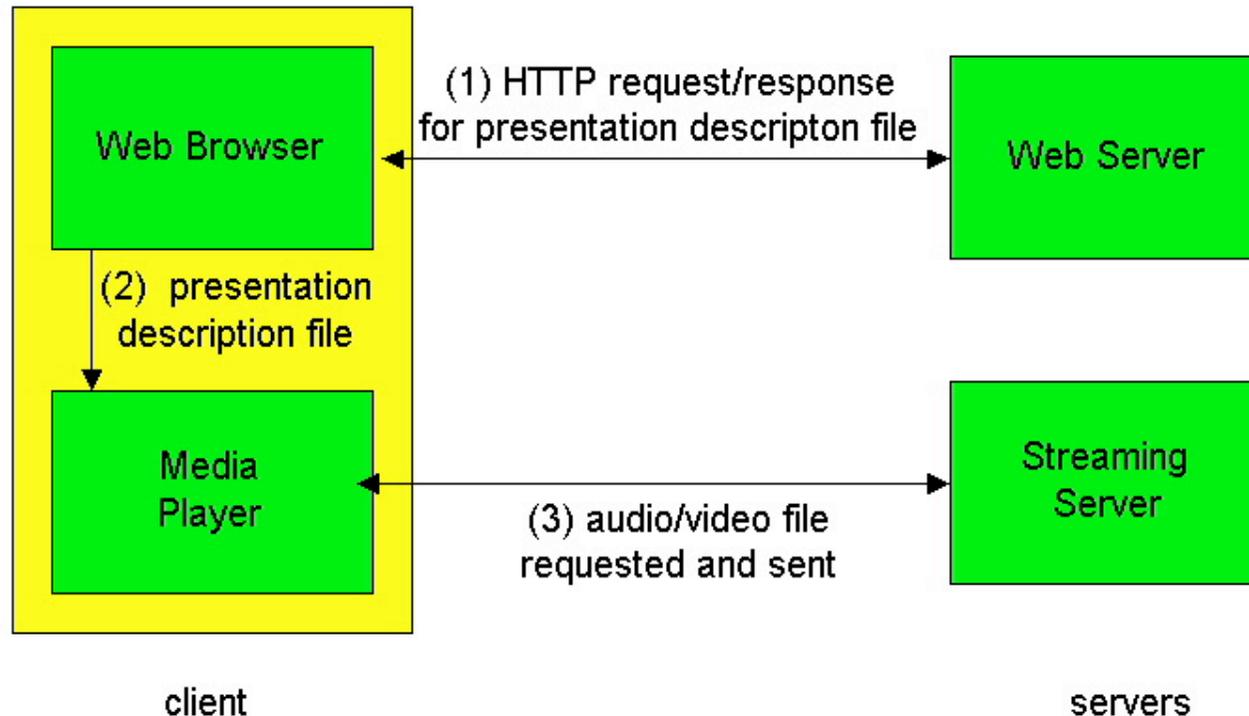


client

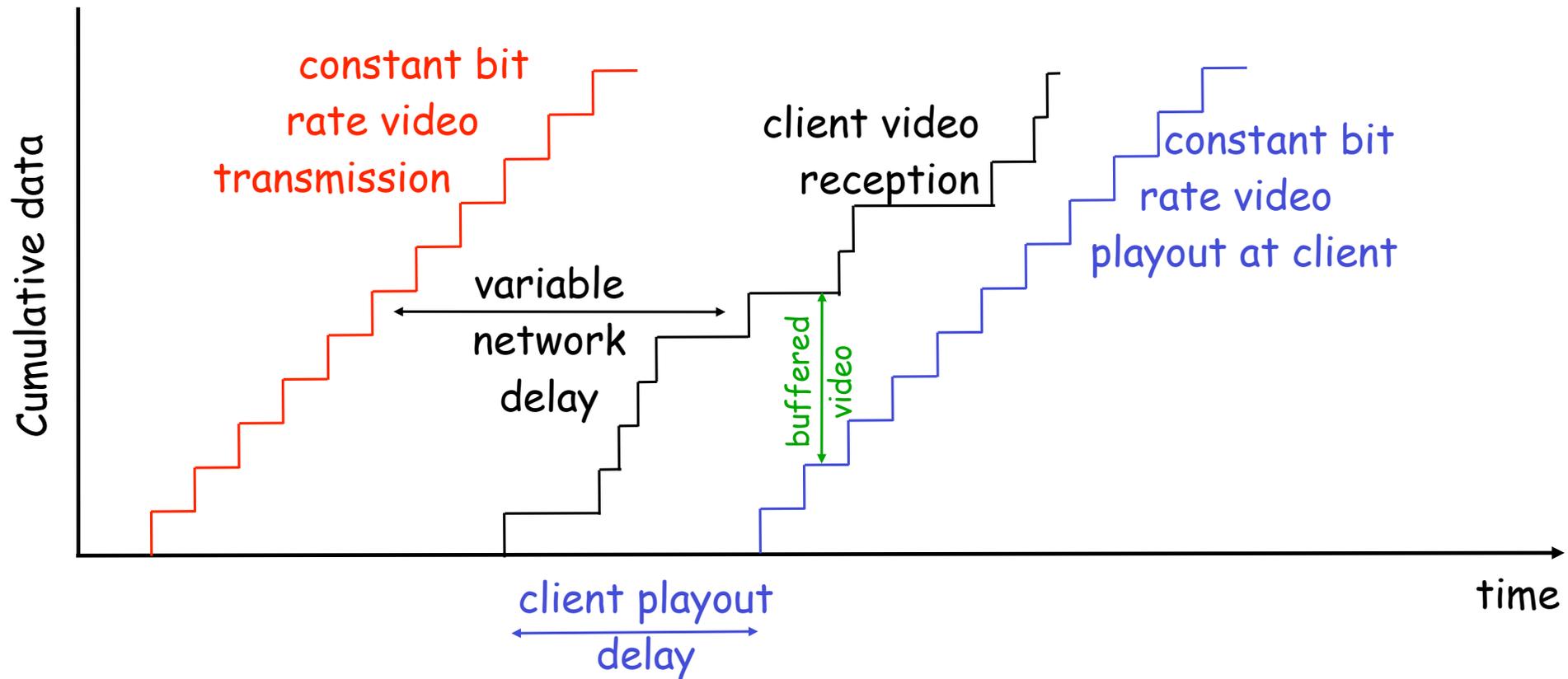
- Audio in Datei gespeichert
- Datei wird als HTTP-Objekt übertragen
 - erst vollständig empfangen
 - dann an den Player weiter gegeben
- Kein Audio oder Video-Stream
- Kein Pipelining
- Extrem lange Delay bis zum Spielbeginn



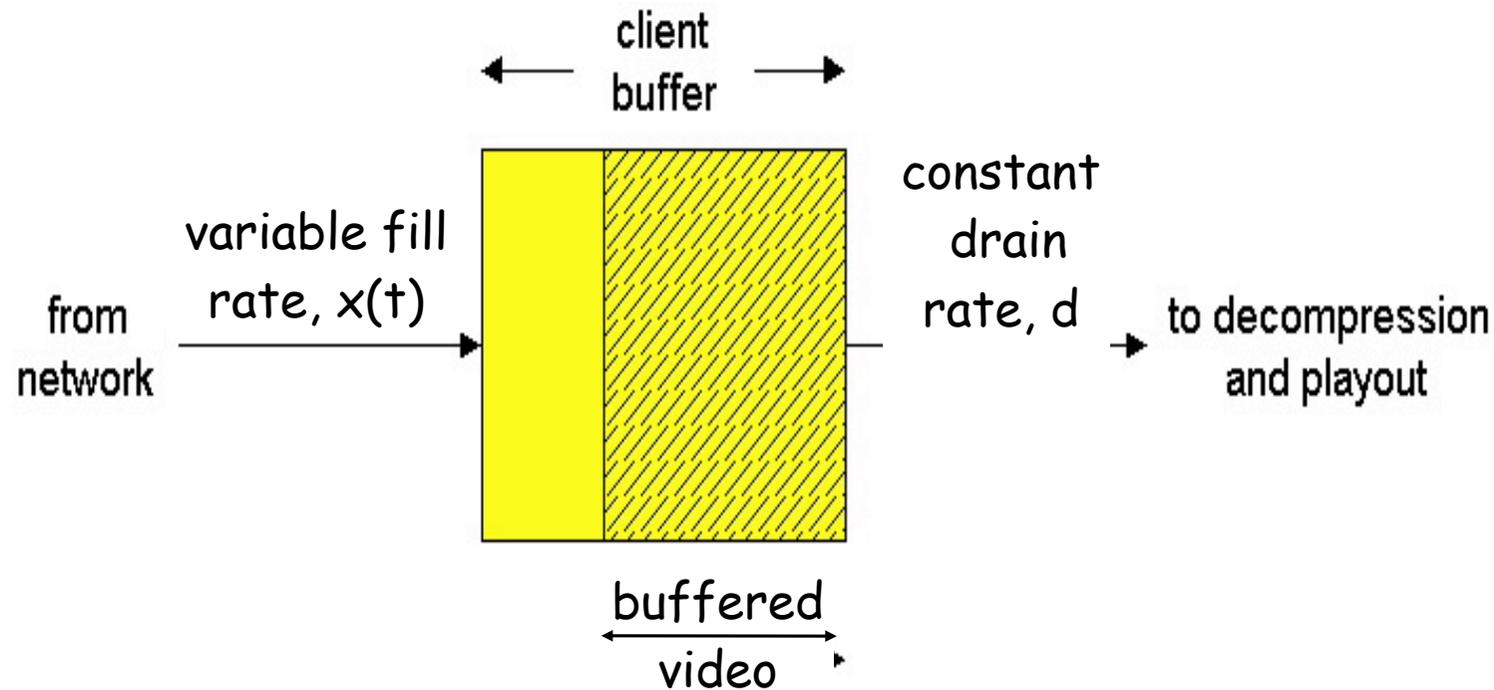
- Browser empfängt Metafile durch GET
- Browser startet Player
 - gibt Metafile weiter
- Player kontaktiert Server
- Server „stream“t Audio und Video zum Player



- Erlaubt Nicht-HTTP-Protokoll zwischen Server und Media-Player
- UDP oder TCP Medienübertragung (3)



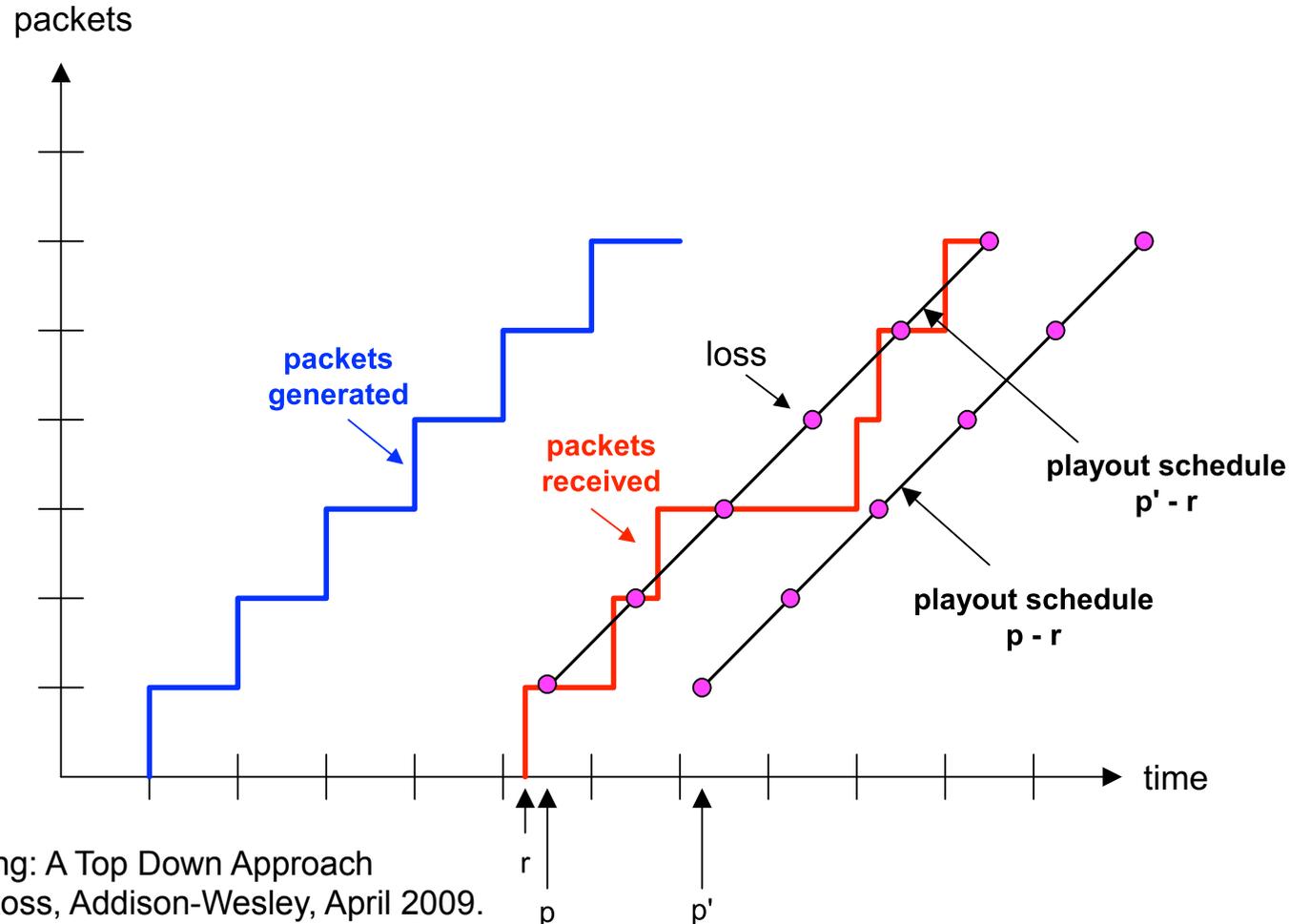
- Puffer auf der Client-Seite
 - Abspielverzögerung kompensiert Netzwerk-Delay und Delay-Jitter



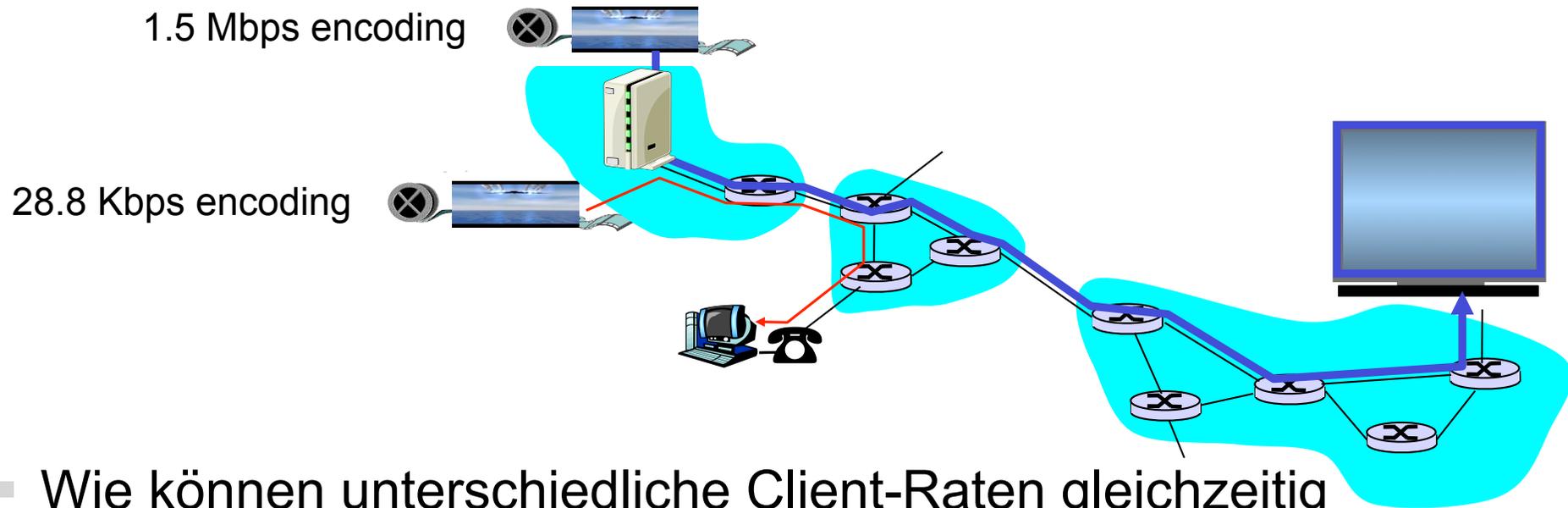
- Puffer auf der Client-Seite
 - Abspielverzögerung kompensiert Netzwerk-Delay und Delay-Jitter

Fixed Playout Delay

- Sender erzeugt regelmäßig Pakete
- erstes Paket kommt zum Zeitpunkt r
- erster Abspiel-Zeitpunkt: p
- zweiter Abspiel-Zeitpunkt: p'



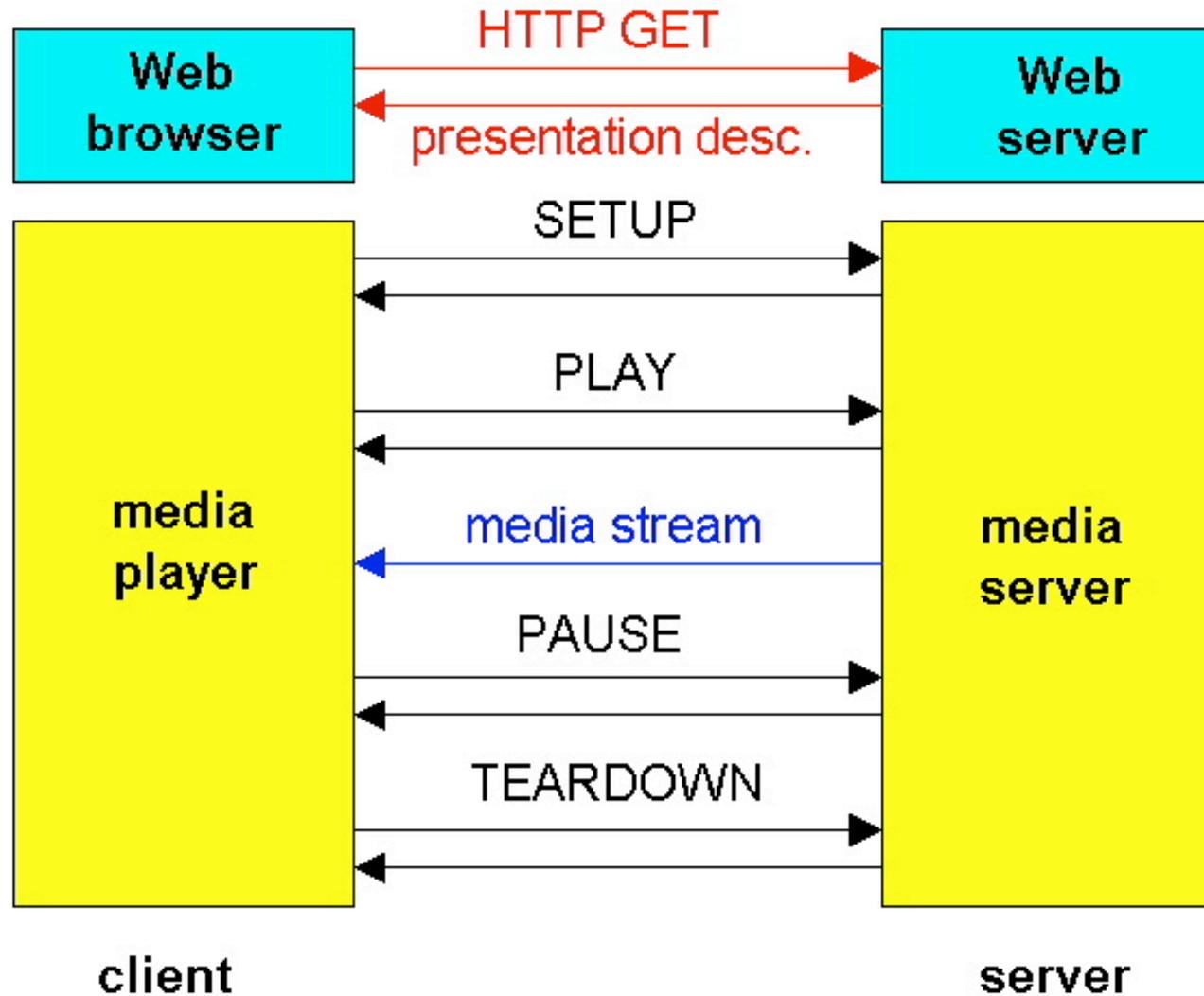
- UDP
- Server sendet an Client angepasste Datenrate
 - unabhängig von der Netzwerklast
 - Senderate = Kodierungs-Rate = konstante Rate
 - Dann ist Fill-Rate = konstante Rate - Paketverlust
 - Kurze Verzögerung für Wiedergabe (2-5 Sek.s) um Netzwerk Jitter zu kompensieren
- TCP
 - sendet auf maximal möglicher Datenrate bei TCP
 - Fill-Rate verändert sich wegen der TCP-Stauvermeidungsstrategie
 - größere Wiedergabe-Verzögerung:
 - gleichmäßig verändernde TCP-Auslieferungsrate
 - HTTP/TCP ist für Firewalls durchlässiger



- Wie können unterschiedliche Client-Raten gleichzeitig bedient werden?
 - 28.8 Kbps Modem
 - 100 Mbps Ethernet
- Der Server speichert und überträgt verschiedene Video-Kopien, die sich in der Kodierungsrate unterscheiden

- HTTP
 - zielt nicht auf Multimedia-Inhalte
 - keine Kommandos für Vorspulen, etc.
- RTSP (Real-Time Streaming Protocol): RFC 2326
 - Client-Server-Protokoll in der Anwendungsschicht
 - Benutzerkontrolle:
 - Rückspulen, Vorspulen, Pause, Resume, Neue-Position, etc...
- Was RTSP nicht leistet
 - keine Definition der Audio/Video-Kodierung über das Netzwerk
 - beschreibt nicht den Transport (UDP oder TCP)
 - beschreibt nicht wie der Media-Player Audio und Video puffert

RTSP Operations

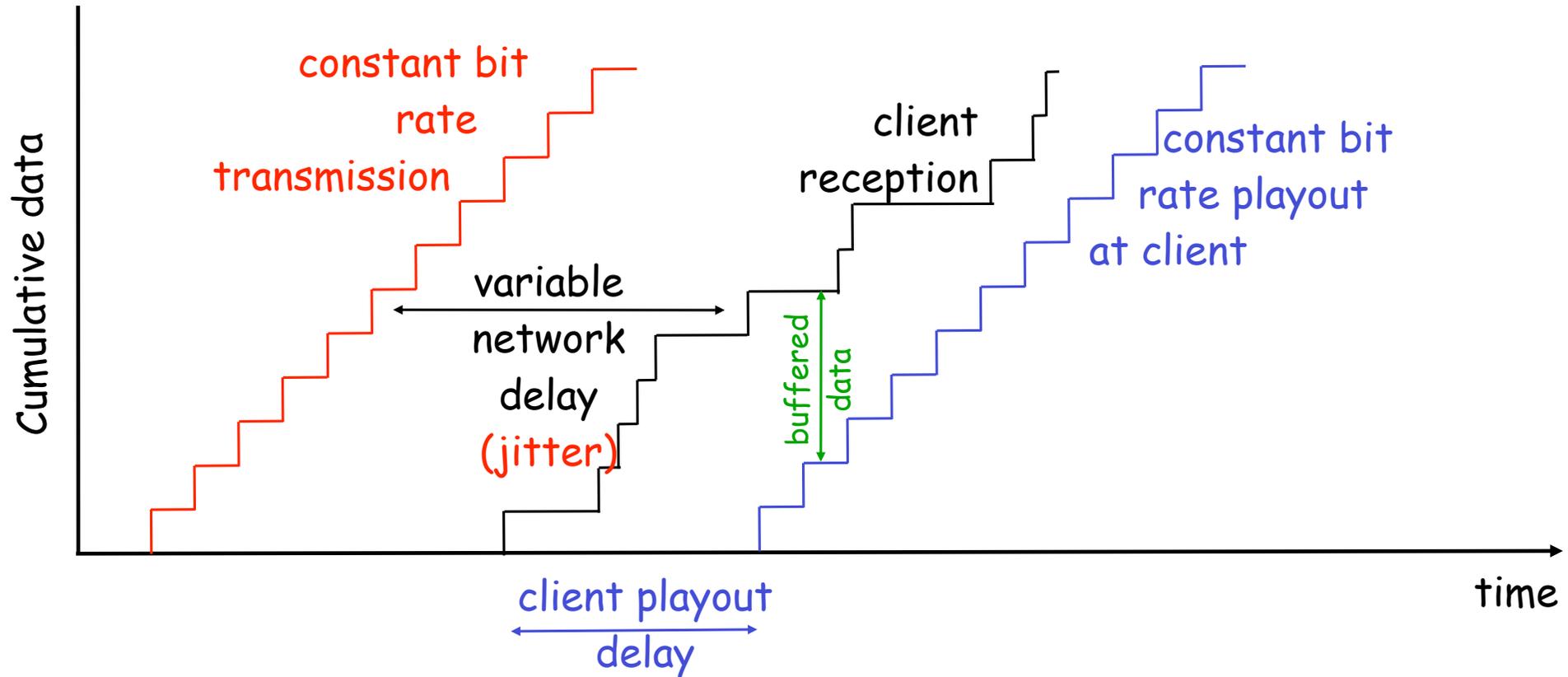


- PC-2-PC phone
 - Skype, iChat, SIP, ...
- PC-2-phone
 - Dialpad, Net2phone, Skype, FaceTime,...
- Video-Konferenz mit Webcams
 - Skype
 - Polycom
 - FaceTime
 - GoogleTalk

- Internet-Telefonie als Beispiel
- Audio: Reden und Schweigen wechseln ab
 - 64 kbps während des Redens
 - Pakete entstehen nur während dessen
 - 20 ms Pakete mit 8 Kbytes/sec: 160 bytes data
- Application Header kommt hinzu
- Daten und Header zusammen in UDP-Paket
- Anwendung sendet UDP Paket alle 20ms während Teilnehmer spricht

- IP-Pakete werden wegen Netzwerkstau verloren
 - Router Puffer läuft über
- Verzögerungs-Verlust (delay loss)
 - IP-Paket kommt zu spät zum Abspielen beim Empfänger an
 - Delays entstehen durch Verarbeitung, Warteschlangen im Netzwerk, Sender und Empfänger-Delays
 - Maximal tolerierter Delay 400 ms
- Verlust-Toleranz
 - hängt von der Stimmkodierung ab
 - Paketverlustraten von 1% bis 10% sind tolerierbar

Delay Jitter

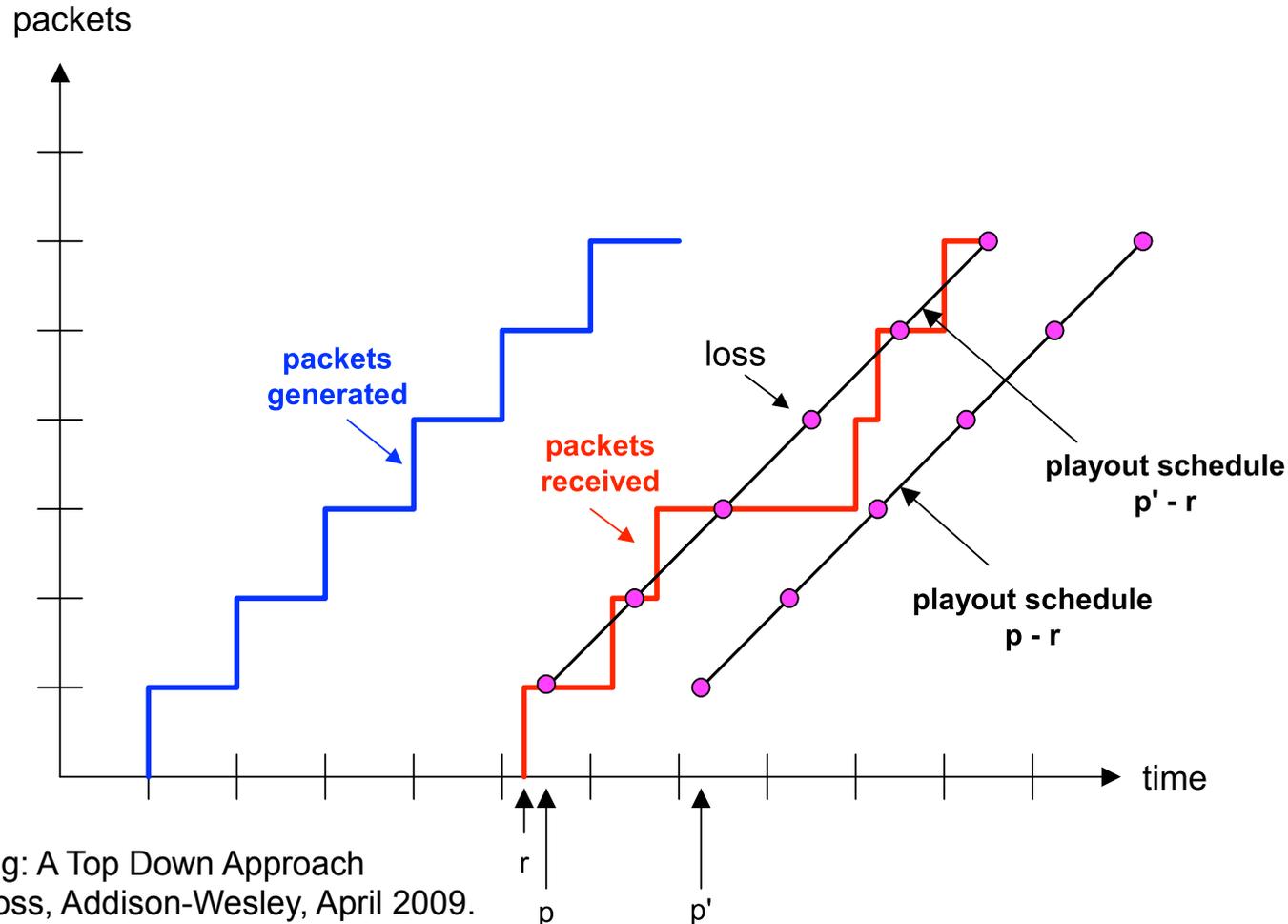


- Gesamtverzögerungen von zwei aufeinander folgenden Pakete kann um 20 ms variieren (transmission time difference)

- Empfänger versucht jedes Paket (chunk) genau q ms nach dessen Erzeugung abzuspielen
 - Chunk mit Zeitstempel t wird zum Zeitpunkt $t+q$ abgespielt
 - Chunks, die nach $t+q$ ankommen, werden übersprungen
- Tradeoff für die Wahl von q :
 - Großes q : seltener Paketverlust
 - Kleines q : bessere Interaktion

Fixed Playout Delay

- Sender erzeugt Paket alle 20 ms wenn ein Tonsignal anliegt
- erstes Paket kommt zum Zeitpunkt r
- erster Abspiel-Zeitpunkt: p
- zweiter Abspiel-Zeitpunkt: p'



- Ziel: minimiere Abspielverzögerung und halte Verlustrate niedrig
- Ansatz: Angepasste Abspiel Verzögerung
 - schätze Netzwerk-Verzögerung und passe die Abspielverzögerung nach jeder Sprechphase an
 - Stille wird verkürzt oder verlängert
 - Pakete werden immer noch all 20 ms während Sprechzeit gesendet
- t_i = Zeitstempel des i -ten Pakets
- r_i = Zeitpunkt, wenn Paket beim Empfänger ankommt
- p_i = Zeitpunkt, wenn Paket abgespielt wird
- $r_i - t_i$ = Netzwerkverzögerung (network delay)
- d_i = Schätzung der Netzwerkverzögerung nach dem Empfangen des i -ten Pakets
- Dynamischer Schätzwert beim Empfänger
 - $d_i = (1-u) d_{i-1} + u (r_i - t_i)$
 - für Konstante, z.B. $u = 0,01$

- Abschätzung für die mittlere Abweichung
 - $v_i = (1-u) v_{i-1} + u |r_i - t_i - d_i|$
- Für das erste Paket ist die Abspielzeit
 - $p_i = t_i + d_i + K v_i$
 - für eine positive Konstante K
- Die Abspielzeit der anderen Pakete ergibt sich daraus

- Wie erkennt man die Sprechzeit
- Ohne Paketverlust ergibt sich das aus den Zeitstempel
 - Falls die Zeitdifferenz folgender Pakete größer ist als 20ms, dann beginnt die Sprechzeit einer neuen Redesequenz
- Ohne Verlust muss der Empfänger auf Zeitstempel und Sequenznummern achten
 - Falls Zeitdifferenz größer ist als 20 ms und Sequenznummern kontinuierlich, dann beginnt die Redesequenz

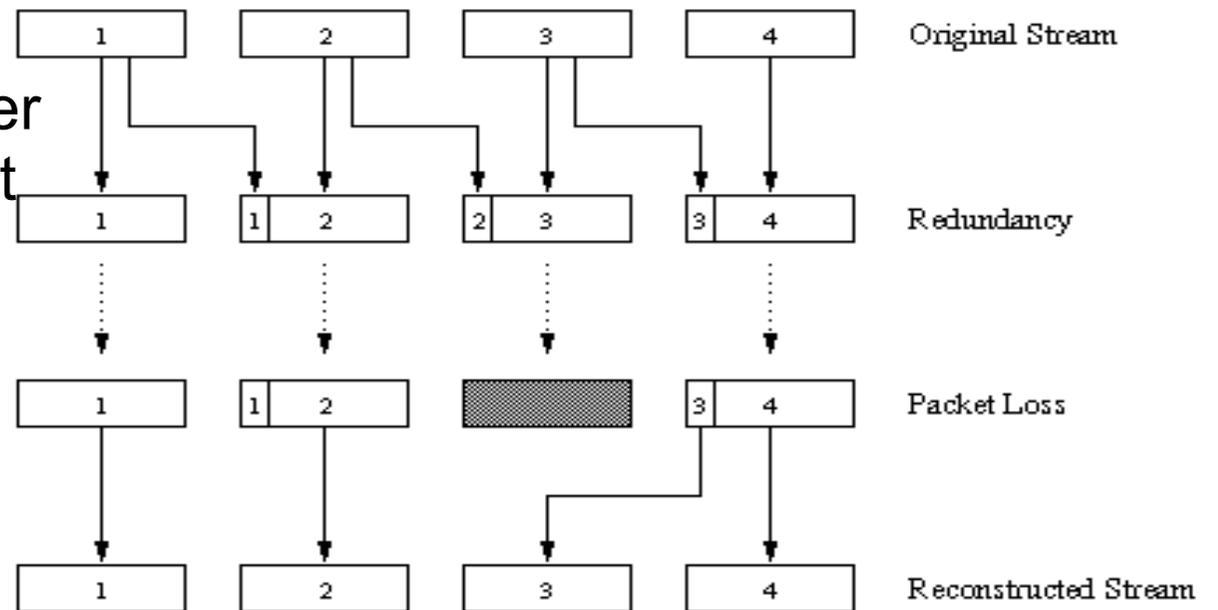
- Forward Error Correction (FEC)
 - für jede Gruppe von n Paketen werden durch Xor ein neues Paket erzeugt
 - sende diese $n+1$ Pakete (Bandweite wird um $1/n$ erhöht)
 - beliebige n Pakete können benutzt werden um ein verlorenes Paket zu rekonstruieren
- Abspielverzögerung
 - muss groß genug sein um alle $n+1$ Pakete zu erhalten
- Tradeoff:
 - größeres n erzeugt kleineren Bandweitenverlust
 - größeres n erhöht Abspielverzögerung
 - größeres n sorgt dafür dass u.U. $n-1$ Pakete verloren gehen

- Zweites FEC Schema

- Audio-Strom mit geringerer Qualität wird mit geschickter Redundanz

- Niedrigere Auflösung liefert redundante Information

- z.B. nominaler PCM-Strom mit 64 kbps und redundanter Strom mit 13 kbps.



- Sobald es einen nicht zusammenhängenden Verlust gibt, kann der Empfänger den Verlust verbergen

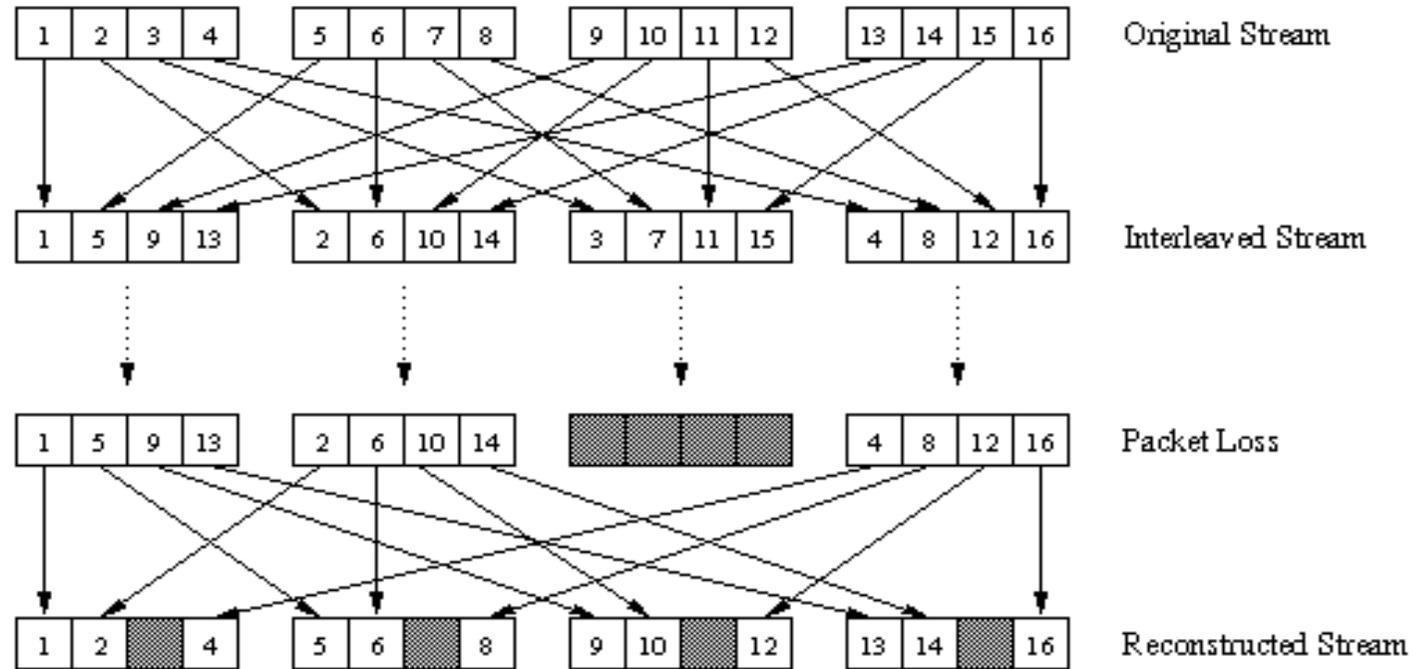
- Interleaving

- Pakete werden in kleinere Einheiten unterteilt
- z.B. Einheiten von 5ms pro Paket

- Pakete enthalten kleinere Einheiten von verschiedenen Paketen

- Bei Paketverlust hat man immer noch die meisten Einheiten von jedem Paket

- Keine Erhöhung der Redundanz



- Verwendung von UDP statt TCP um durch Stauvermeidung erzeugte Verzögerungen für zeitnah zu versendende Pakete zu vermeiden
- Empfängerseitig angepasste Abspielverzögerung um Verzögerung zu kompensieren
- Serverseitig wird die Strom auf die zur Verfügung stehende Bandweite zum Empfänger angepasst
 - Auswahl aus vorkodierten Raten
 - Dynamische Server-Kodierungsrate
- Verlustverarbeitung (Anwendungsschicht wegen UDP)
 - FEC, Interleaving, Fehlerkaschierung
- Content Distribution Networks
 - Inhalt wird in die Nähe des Empfängers gebracht

Systeme II

8. Multimedia

Christian Schindelhauer

Technische Fakultät

Rechnernetze und Telematik

Albert-Ludwigs-Universität Freiburg