Übung zur Vorlesung

Multimedia im Netz

Ludwig-Maximilians-Universität
Wintersemester 2010/2011

Übungsblatt 10

- Audio Streaming mit TCP

- Programm wird über Kommandozeile gestartet
  - java AudioServer
  - java AudioClient „Adresse“ „Port“

- Server:
  - Audioeingang über Mikrofon
  - streamt die aufgenommenen Audiodaten an den Client

- Client
  - verbindet sich mit dem Server
  - hört dadurch, was dort ins Mikrofon gesagt wird

Warum Audio mit TCP

- Für das menschliche Gehör ist (mäßiger Paketverlust) kein Problem und UDP verursacht weniger Overhead bei der Übertragung und ist dadurch schneller

  → UDP wird meist für Audio- und Videostreaming verwendet

- ABER: Wenn man UDP für die Audioübertragung verwendet, müssen Zeitstempel oder Paketnummern verwendet werden, um:
  - die Reihenfolge zu kontrollieren
  - Paketverlust festzustellen
  - duplizierte Pakete zu erkennen

→ TCP übernimmt das für uns

Audio Qualität vs. Datenrate

	SampleRate (Hz)	Bits per Sample	Mono/Stereo
Telefon	8000	8	Mono
Radio	22050	16	Stereo
CD	44100	16	Stereo

→ Für unsere Zwecke orientieren wir uns am Telefon

Audio aufnehmen (1)

- `Line`: Interface, das mit mono- und mehrkanal Audioströmen umgehen kann; Wird sowohl für Audioeingang als auch -ausgang verwendet

- `DataLine.Info`: Fügt der `Line` verschiedene Funktionalitäten hinzu
  Zum Beispiel: `start`, `stop`, `drain` oder `flush`

- `TargetDataLine`: Spezielle `DataLine` von der Audio gelesen werden kann. Daten kommen hier zum Beispiel von einem Mikrophon

  - `open(AudioFormat format)`: Öffnet die Line und reserviert alle nötigen Resourcen

  - `start()`: Startet die tatsächliche Aufnahme
    (Hinweis: Methode von DataLine, nicht von TargetDataLine)

Audio aufnehmen (2)

- In den `OutputStream` schreiben:
  - `read(byte[] b, int off, int len)`: Liest Audiodaten in den InputBuffer der Line
  - `write(byte[] b, int off, int len)`: Schreibt die Daten auf den Outputstream

Audio abspielen (1)

- `InputStream` zu `AudioInputStream` aufwerten
- Analog zur Aufnahme: `DataLine.Info` und statt `TargetDataLine` eine `SourceDataLine`
- `SourceDataLine`: Ist eine spezielle `DataLine` auf die Audio geschrieben werden kann. Sie buffert die Datenbytes wenn nötig
  - `open(AudioFormat format)`: Öffnet die Line und reserviert alle nötigen Resourcen
  - `start()`: Startet das Abspielen
    (Hinweis: Methode von Dataline und nicht SourceDataLine)

Audio abspielen (2)

- Aus dem AudioInputStream lesen
  - `read(byte[] b, int off, int len)`: Liest Audiodaten in den InputBuffer der Line
  - `write(byte[] b, int off, int len)`: Schreibt die Daten aus dem Buffer auf die line und gibt sie dadurch als Ton aus
  - `drain()`: Sorgt dafür, dass der Buffer bis zum Ende ausgelesen wird.

Beispiel