Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Kommunikation und Multimedia
Prof. Dr. L. Wolf

Praktikum Kommunikationssysteme im SS06

– Aufgabenbeschreibung –

Betreuer: Zefir Kurtisi, Habib-ur-Rehman, Alexej Beresnev

Aufgabe 2: Voice Chat Tool (VCT)

Termine

Konzept	01.06.06
Abgabefrist	29.06.06

1 Einleitung

In der ersten Aufgabe wurde für die Kommunikation innerhalb der IBRC-Anwendung auf das verbindungsorientierte und zuverlässige TCP gesetzt. In dieser Aufgabe wird es um den Transport von Echtzeitdaten gehen, für das sich TCP wenig eignet.

Es soll eine einfache Anwendung implementiert werden, die zwei Teilnehmern eine netzwerkgestützte Sprachkommunikation ermöglicht. Da es sich bei Sprache um sehr zeitkritische Daten handelt, ist eine Datenübertragung über UDP inklusive zugehöriger Maßnahmen zur Flusskontrolle, Fehlererkennung und -Behandlung zu implementieren.

1.1 Hintergrund

Paketbasierte Sprachübertragung ist derzeit dabei, alle bislang eingesetzten Verfahren zu ersetzen: während das Mobilfunknetz der dritten Generation (UMTS) ausschließlich paketbasiert arbeitet, preschen DSL-Anbieter mit integrierten VoIP-Angeboten auch in den privaten Bereich vor. Konnten bislang nur ,,Eingeweihte" mit ihren Rechnern günstig über das Internet telefonieren, wird die Technik für den Benutzer nicht sichtbar in Consumer-Endgeräte integriert.

Die Sprachqualität, die in Form von geringer Verzögerung und kontinuierlichem, störungsfreien Audio wahrgenommen wird, wurde auf akzeptable Niveaus gehoben – die Technik an sich bleibt jedoch anspruchsvoll. Diese praktisch nachzuvollziehen ist Ziel dieses Teils der Aufgabe.

1.2 Anforderung an VoIP-Übertragung

Die Übertragung von Echtzeit-Audiodaten stellt die höchsten Anforderungen an die Kommunikationstechnik. Das liegt an der Sensitivität des menschlichen Gehörs, das Übertragungsfehler sehr leicht erkennt und als unangenehm empfinden lässt. Im Vergleich dazu ist die Toleranz des Sehsinns deutlich höher, so dass bei Anwendungen wie der Videotelefonie die Audioübertragung immer höher priorisiert wird als das Video.

Drei Faktoren beeinträchtigen die wahrgenommene Qualität von paketbasiert übertragenen Audiodaten:

Paketverlust
: Der Verlust eines Audiodaten-Paketes verursacht eine Diskontinuität im Datenstrom, der sehr deutlich, meist als Knacks, wahrgenommen wird. Paketverluste treten primär auf, wenn Pakete auf ihrem Weg vom Sender zum Empfänger von einem überlasteten Router verworfen werden, oder auf drahtlosen Kanälen verloren gehen. Bei interaktiven Echtzeit-Daten müssen zusätzlich die Pakete als Verlust gewertet werden, die nicht rechtzeitig ankommen und verworfen werden müssen.

Verzögerung
: Bei der Verarbeitung von digitalem Audio ist eine systembedingte Verzögerung nicht zu vermeiden: die Soundkarte führt am Eingang eine Analog-Digital- und am Ausgang eine D-A-Umwandlung durch, die über mehrere Buffer-Stufen mit dem Betriebssystem kommuniziert. Dort erfordern Interruptbehandlungen und Netzzugriff ebenfalls eine gewisse Zeit, so dass eine minimale Verzögerung von bis zu 20 ms nicht verhindert werden kann. Um die oben beschriebenen Paketverluste durch ein erneutes Senden korrigieren zu können, werden zusätzliche Buffer vor dem Abspielen eingeführt. Die Gesamtverzögerung aus System-Latenz und Streaming-Buffer sollte in einem interaktiven System laut [2] 120 ms nicht überschreiten.

Verzögerungsschwankung (Jitter)
: Eine Übertragungsverzögerung ist zwar störend, das Gehör ist jedoch in der Lage sich anzupassen, solange diese konstant ist. Starke Schwankungen wirken sich einerseits negativ auf die Wahrnehmung aus, und verursachen andererseits technische Probleme wie Buffer-Über- und Unterläufe. Ein Jitter kann vermieden werden, wenn ein Buffer angemessener Länge eine konstante Verzögerung vorhält, während der Fehler korrigiert werden können.

2 Aufgabenstellung

Es soll eine Anwendung implementiert werden, die eine paketbasierte Sprachkommunikation zwischen zwei Rechnern ermöglicht. Das zugrunde liegende Prinzip ist dabei relativ einfach: für den Hinweg werden die Daten der eigenen Audiohardware abgegriffen, paketisiert und per UDP zum Zielrechner gesendet. In der Gegenrichtung werden Datenpakete empfangen, serialisiert und über die Audiohardware ausgegeben.

Etwas verkompliziert wird dieses Grundprinzip durch die oben erwähnten strikten Anforderungen nach möglichst fehlerfreier Datenübertragung sowie geringer Verzögerung und Jitter.

2.1 Audio-Schnittstelle `pa_interface`

Die Schnittstelle zur Audiohardware bildet beim OS/X das *CoreAudio*, das sehr umfangreiche Zugriffsmöglichkeiten bietet und mit einer entsprechend großen API zu bedienen ist. Da die Einarbeitung und die Implementierung des Zugriffs auf das Audio den zeitlichen Rahmen für diese Aufgabe sprengen würde, wird für diese Aufgabe ein Programmgerüst vorgegeben.

Dieses Interface setzt auf die plattformunabhängige *PortAudio* [1] Bibliothek auf. In der Datei `pa_interface.c` sind die Zugriffe über diese Bibliothek implementiert und in der Headerdatei `pa_interface.h` definiert. Im Programmgerüst ist in der Datei `vct.c` beispielhaft die Nutzung der Schnittstelle implementiert.

2.2 Ziel

Ein Aufruf von `make` im bereitgestellten Programmgerüst erzeugt die Datei `vct`, deren Aufruf die vom Mikrofon eingelesenen Audiodaten am Lautsprecher ausgibt, also den Audio-Eingang mit dem -Ausgang kurzschliesst. In der zu implementierende Aufgabe sollen dagegen die Audio-Eingangsdaten der beiden miteinander kommunizierenden Rechner am jeweils anderen ausgegeben werden.

Das Programm `vct` soll dabei mit bis zu drei Parametern aufgerufen werden können:

`./vct` $< host > [< port >][< buffer\_delay >]$

Die Werte *host* und *port* sind selbsterklärend, der Port sollte einen Defaultwert haben. Das `buffer_delay` soll die Verweildauer der empfangenen Pakete in Millisekunden sein, bevor diese ausgegeben werden, für dieses Wert soll ein Defaultwert von 100 ms angenommen werden. Das Buffering ist so zu dimensionieren, dass eine entsprechende Anzahl von Paketen vorgehalten werden kann. Sobald dieses Empfangsbuffer erstmalig bis zur angegebenen Länge gefüllt ist, startet der Rechner die Audioausgabe und sorgt fortan für einen kontinuierlichen Datenfluss zur Audiohardware. Die Ausgabe wird abgebrochen, wenn der Empfangsbuffer einmal komplett unterläuft, was ein Indiz dafür ist, dass der Gesprächspartner keine weiteren Pakete mehr sendet.

Es gibt keine Vorgaben, was Kommunikation (Protokoll, Paketgrößen usw.) und Umsetzung (Buffering Strategien, Sequenzierung, Dejitter usw.), es sind nur die nachfolgenden Anforderungen einzuhalten.

2.3 Anforderungen

Hinsichtlich der genannten Grenzen und Schwierigkeiten bei der Implementierung einer VoIP-Anwendung herrschen innerhalb des LANs nahezu perfekte Bedingungen vor: die Übertragungsverzögerung ist nahezu Null, der Jitter ebenfalls minimal und auch Paketverluste wird man hier nicht allzu häufig antreffen.

In dieser Umgebung ist eine Umsetzung mit einer konstanten Gesamtverzögerung (Ende-zu-Ende) von weniger als 100 ms zu implementieren. Dieses ist die Mindestvoraussetzung, mit den Aufrufparameter `buffer_delay` soll darüberhinaus der minimale Wert ermittelt werden, bei dem die Paketverlustrate durch den Jitter 5 % nicht übersteigt. Der Datenfluss von und zur Audio-Hardware muss kontinuierlich sein, um fehlerfreie Wiedergabe zu ermöglichen. Das kann dadurch gewährleistet werden, dass an die Audio-Hardware jeweils gleich viel Daten geschrieben, wie von dort gelesen werden.

In falscher Reihenfolge eintreffende Pakete müssen umsortiert werden, entsprechende Bildschirmausgaben sollen diese Ereignisse dokumentieren. Paketverluste und Pakete, die nach ihrem Abspielzeitpunkt eintreffen, sind ebenfalls als solche zu dokumentieren. Zusätzlich sind zur besseren Wahrnehmung für verlorenen Pakete Audio-Blöcke entsprechender Dauer an die Audiohardware auszugeben, die ein akkustisches 1kHz-Signal erzeugen.

2.4 Funktionen für den Audiozugriff

Die Parameter, mit denen das Audio-Device initialisiert wird, sind in `pa_interface.h` zu finden. Die vorgegeben Werte sind eine Abtastrate von 32 kHz bei einer Auflösung von 16 Bit pro Sample und zwei Kanälen. Damit hat jedes Sample 4 Byte und es werden 128 kByte pro Sekunde erzeugt und verbraucht.

In der gleichen Header-Datei sind auch die Funktionen definiert, die zum Zugriff auf PortAudio vorgesehen sind:

`int init_portaudio(void)`
: Öffnet PortAudio im blocking-IO Modus mit den oben genannten Parametern. Muss aufgerufen werden, bevor weitere Zugriffe auf PortAudio erfolgen dürfen.

`void close_portaudio(void)`
: PortAudio-Stream schließen; sollte vor dem Verlassen des Programms aufgerufen werden.

`int get_pa_output_samples(void)`
: Gibt die Anzahl der Samples zurück, die in die Audio Ausgangsbuffer geschrieben werden können.

`int get_pa_input_samples(void)`
: Gibt die Anzahl der Samples zurück, die aus dem Audio Eingangsbuffer gelesen werden können.

`int read_audio(char *buff, int size)`
: Liest aus dem Audio-In-Buffer maximal `size` Bytes nach `buff`, der nicht `NULL` sein darf. Liegen keine Daten vor, ist der Rückgabewert Null, ansonsten wird die Anzahl der tatsächlich gelesenen Bytes zurückgegeben.

`int write_audio(char *buff, int size)`
: Gegenstück zum Schreiben von Daten in dem Audio-Out-Buffer. Ein unvollständig geschriebenes Paket (Rückgabewert ungleich `size`) weist auf Probleme mit der Synchronisation hin, da versucht wird, mehr Daten zu schrieben als gelesen wurde.

3 Abnahme

Für diese Aufgabe stehen vier Wochen Bearbeitungszeit zur Verfügung, wobei eine Woche für die Ausarbeitung des Konzeptes vorgesehen ist, das in der ersten Woche abzugeben ist.

Die Aufgabe gilt als abgenommen, wenn das Konzept eingereicht und die darauf basierende Implementation die genannte Funktionsfähigkeit nachweist. Der Quellcode wird vorab zur Prüfung per Email an die Betreuer gesendet. Während der betreuten Zeiten wird dann das Programm auf den Rechnern im G40 demonstriert und besprochen. Bei dieser Abnahme müssen alle Gruppenmitglieder anwesend sein, um den eigenen Programmteil erläutern und Fragen zur Implementation beantworten zu können.

4 Hinweise

Die iMacs verfügen leider nicht über einen Vorverstärker am Mikrofoneingang, so dass handelsübliche Headsets nicht eingesetzt werden können. Das eingebaute Mikrofon ist jedoch ausreichend, wenn die Empfindlichkeit hochgeregelt und der Sprecher sich relativ nah vor dem Gerät befindet. Für die Audio-Ausgabe müssen Kopfhörer benutzt werden, da sich sonst Rückkopplungseffekte einstellen. Bei Bedarf stellt das IBR Kopfhörer zur Verfügung.

Literatur

[1] Ross Bencina, Phil Burk, et al. PortAudio - portable open-source cross-platform Audio API. Online: `http://www.portaudio.com`.

[2] R. Steinmetz and K. Nahrstedt. *Multimedia: Computing, Communications and Applications*. Prentice Hall, Upper Saddle River, 1995.