

Adaptive Real-Time Interactive Mediaphone CS641 Project

Weichao Wang
Yuhui Zhong
Dec 06/2002

1 Motivation

The project is to implement a PC-to-PC network video phone. In this project, we try to integrate real time audio and video transformation. The data are transferred through RTP/UDP. The system does not support broadcast or multicast, so the “phone call” will be point to point. Because the available bandwidth for the network connection fluctuates a lot, the system should have dynamic adjustment to guarantee audio and video quality.

With the implementation, we can test different kinds of audio/video codec functions and the bandwidth they consume. It can also test their reflection to bandwidth fluctuation. If a software router is inserted in the middle of the link, it can test the system response to packet drop and out of order. Different kinds of adaptable methods have been designed for the real time traffic on the internet. Using the system as a platform, other developers can easily examine their new mechanisms on the real application. The system can also be used for the interactive multimedia database query system with minor change. In that application, the bandwidth usage will not be balanced. The multimedia file transferred from the database to the end user will consume much more resource. The implemented system can examine the response time and performance for the service.

2 Implemented function and referenced code

We have implemented the following components of the code:

- Class structure design and functionality definition
- Audio channel design and implementation, G723.1 audio codec
- Video channel design and implementation, H.263 video codec
- Adaptive methods. Silence detection and sound burst in audio channel
- Reduced RTP and RTCP
- Buffer management. Double link list buffer structure and jitter buffer.

The project is a large system. We reference other implementations for the following components of code.

- Video capture driver
- Code for RTP and RTCP
- H.263 motion prediction module

3 Introduction to related protocols and standards

3.1 RTP & RTCP (RFC 1889)

The Real-Time Transport Protocol, or RTP, was designed to send real-time media such as voice and video over UDP/IP, though it can be used to transmit other types of data such as text and pointers. The protocol also supplies information to allow a receiver to re-synchronize the media for lip syncing or having text appear at the correct time in relation to an image or word. Since RTP can be configured for low latency, it is useful for interactive conversations as well as streaming media. RTP streams can be sent to unicast

or multicast destinations. Using multicast allows for bandwidth efficient sessions in which many people are involved such as an online lecture or press conference.

The Real-Time Transport Control Protocol, or RTCP, is a companion protocol to RTP for gathering statistics on a media connection and information such as bytes sent, packets sent, lost packets, jitter, and round trip time. The application can use this information to judge the quality of its connections and make adjustments as required such as changing from a low compression codec to a high compression codec. For security, the RTP/RTCP data can be encrypted to prevent unwanted listeners to a media stream.

3.2 H.263

H.263 is a provisional ITU-T standard. It was designed for low bitrate communication, early drafts specified datarates less than 64 Kbits/s, however this limitation has now been removed. It is expected that the standard will be used for a wide range of bitrates, not just low bitrate applications. It is expected that H.263 will replace H.261 in many applications.

The coding algorithm of H.263 is similar to that used by H.261, however with some improvements and changes to improve performance and error recovery. Half pixel precision is used for motion compensation whereas H.261 used full pixel precision and a loop filter. Some parts of the hierarchical structure of the datastream are now optional, so the codec can be configured for a lower datarate or better error recovery. There are now four optional negotiable options included to improve performance: Unrestricted Motion Vectors, Syntax-based arithmetic coding, Advance prediction, and forward and backward frame prediction similar to MPEG called P-B frames.

H.263 supports five resolutions. In addition to QCIF and CIF that were supported by H.261, there are SQCIF, 4CIF, and 16CIF. SQCIF is approximately half the resolution of QCIF. 4CIF and 16CIF are 4 and 16 times the resolution of CIF respectively. The support of 4CIF and 16CIF means the codec could then compete with other higher bitrate video coding standards such as the MPEG standards.

3.3 G723.1

G723.1 is an ITU-T standard for audio compression algorithm. It usually needs the bandwidth for 5.3 to 6.3 kbps. Compared to the standard PCM audio sample rate (8000 sample per second, 16 bits per sample), the algorithm saves 96% of the bandwidth. G723.1 has become a standard for the Microsoft product and is widely applied.

4 Main Graph

The system supports continuous media via RTP/UDP connections. The network latency detection and jitter buffer mechanism are applied to smooth the flow of data during bandwidth fluctuation. Both RTCP and ICMP are used to detect the transfer delay. We put the audio data with higher priority because it is more important for the real-time network conference system. The class structure is shown in page 9.

We will briefly describe the following classes and their functionalities.

4.1 Class EndPoint

Endpoint is an abstraction of a user in a session. It encapsulates the user's information, the host information, and video/audio codec. The most important thing is that it maintains the capability table, which records the codec ability of this node. This information is useful during session establishment period. The users can choose suitable codec between them so they can talk to each other.

4.2 Class Capacity

The class capability records the encode/decode ability of an entity in the session. It can be divided into 4 groups: Audio, Video, Data, and User Input. The audio codec includes G.722, G.711, and G.723.1. We choose G723.1 in our implementation because of its simplicity and bandwidth efficiency. The video capability in our system is H.263. Two kinds of frame size: QCIF (176 * 144 pixel) and SQCIF (128 * 96 pixel), are supported. The data capability is reserved for the text interaction such as white board editing. And the User input is reserved for other applications and further extension.

4.3 Class Codec

Class codec records the specific codec ability, and the main operations it has are read and write:

Read(): get data from capture device buffer, encode it, and put it into send buffer list;

Write(): get data from jitter buffer or receive buffer, decode it, and play on device;

4.4 Class Connection

In a session, the users establish a group of logical and physical connections. This class is the collection of them. It also helps user to finish connection establishment and bandwidth allocation;

4.5 Class RTP Session

In this project, the audio/video data are sent through RTP protocol. The RTP session is in charge of RTP packet encapsulation, send and receive. It also maintains information for sequence order and time interval;

4.6 Class Channel

It is the logical channel in the session. Its primary functions include transmit and receive;

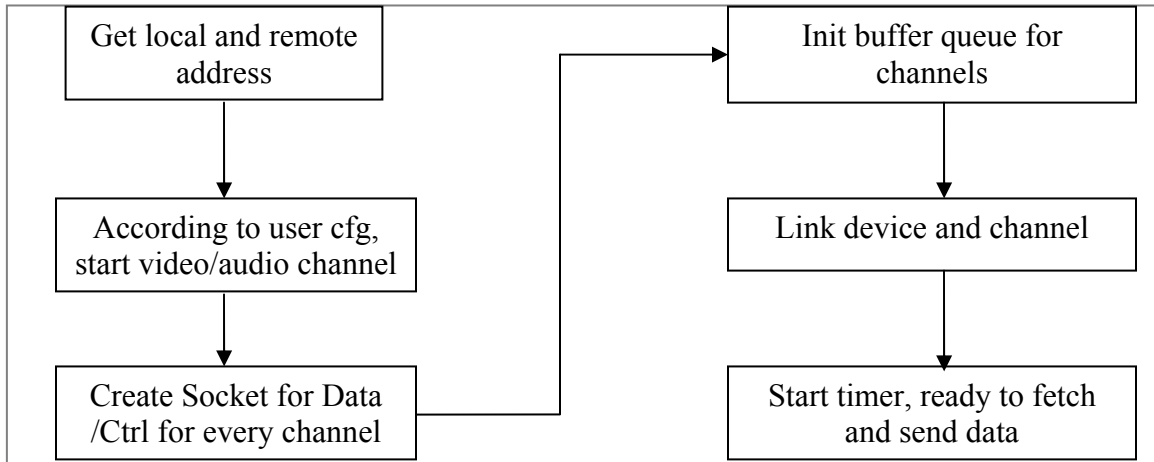
Transmit(): keeps reading data from device, encode them and send out;

Receive(): keeps reading data from network, decode them and write into device buffer;

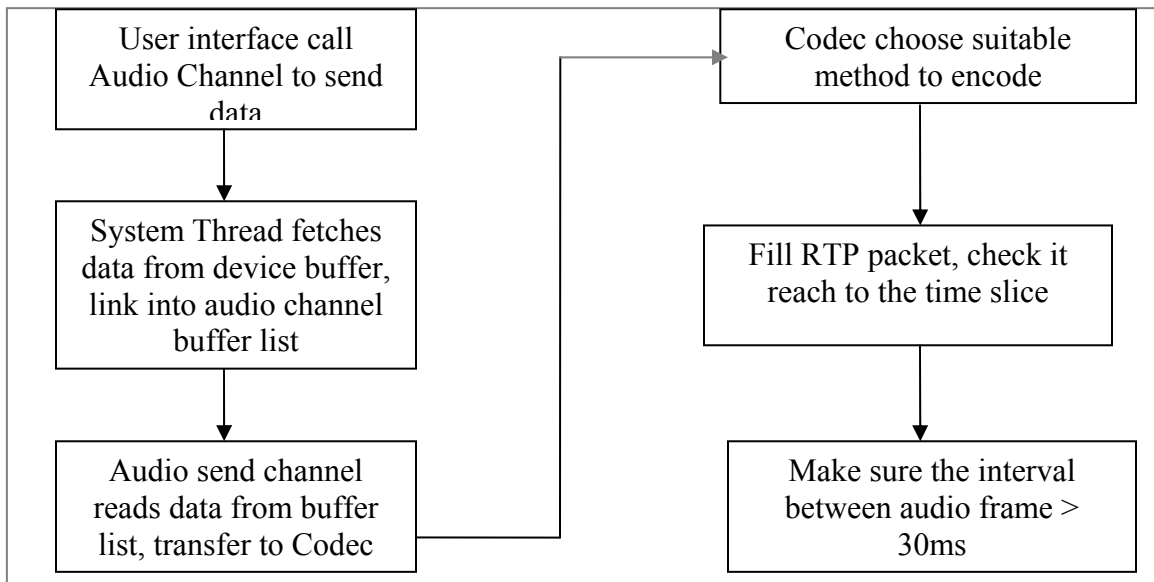
4.7 Typical Module Introduction

The following figures show some of the primary procedures in the system

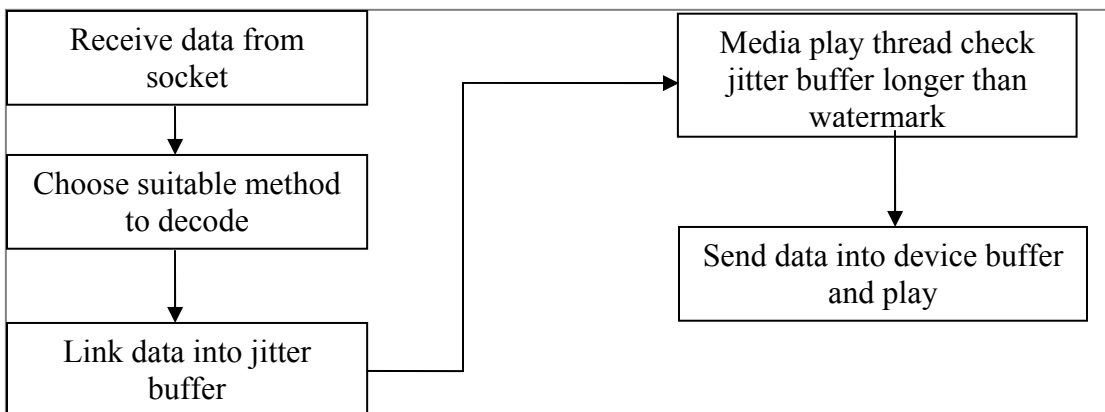
•EndPoint Get Started



•Audio on Send



•Audio on receive



5 Silence Detection function for Audio

The user of the system does not keep on talking during the whole session. Detecting the silent period and stopping to send data will save the bandwidth. The threshold of noise can be predefined or dynamically determined. In our system, a dynamic adjustment mechanism is applied. Every end will maintain two parameters, which identify the minimum voice power strength and the maximum noise power strength. We must be aware that the noise and voice intervals can overlap. In our implementation the unit time of voice detection is 30 ms. The number of continuous silent or voice slot that can flip the detection state is predefined. The steps can be summarized as:

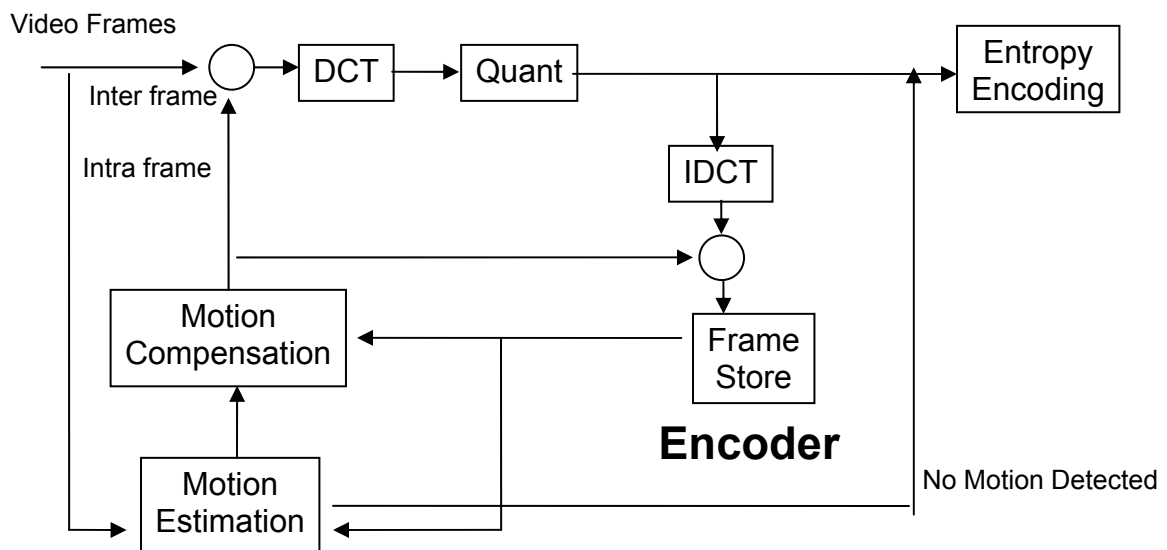
- Sender calculate the Threshold, if enable detection, data will be sent only when speaking;
- The sound signal are not uniformly speed streaming, have pause and burst;
- one RTP packet can contain multi sound slot;
- if state not change(still speak, or still silent), the packet can be sent on uniform speed;
- if change from speaking --> silence, even the current RTP packet is not full, send it out;
- on receive end, if silence, play some background noise;

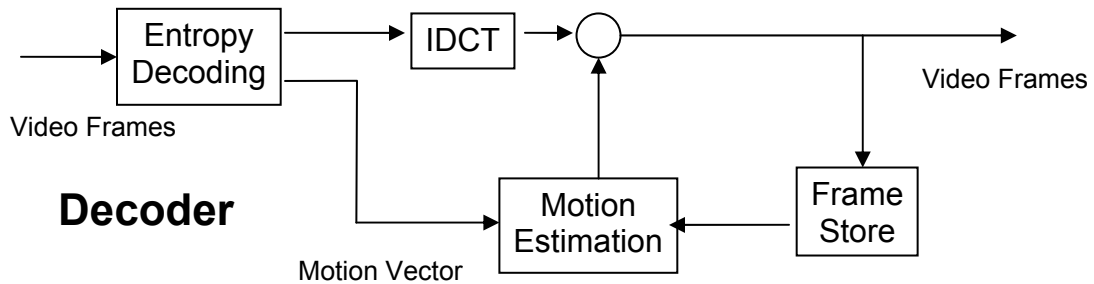
6 Video Channel

The coding for video is based on H.263. One important thing about our system is that we do not implement any mechanism to synchronize video with audio phase. There are some algorithms available but we run out of time.

The primary functions of the video channel include:

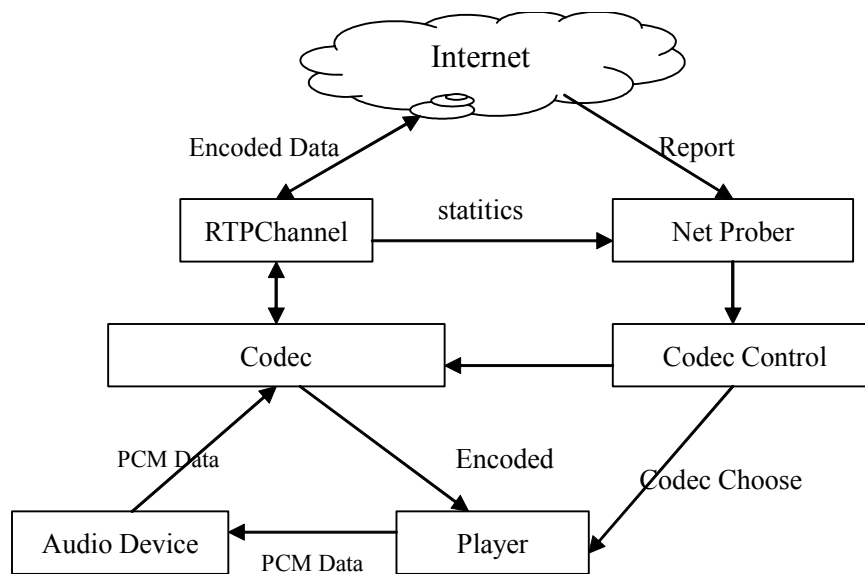
- Init(): create capture window, link capture with device driver, open preview, set call back function for display buffer;
- VideoCallbackProc(): link newest display buffer into list, need to implement mutex, if no free buffer left, discard the oldest frame and reuse the buffer
- Read(): return a frame in RGB format, need to maintain free list and data list in mutex mode;





7 Audio Channel

In network conference, audio is more important than video. Three methods can be applied to guarantee audio quality: jitter buffer, adaptive codec method, and Forward Error-Correction (FEC). We choose the jitter buffer because it is easy to implement and debug. The structure of audio channel is shown as:



7.1 Implementation of jitter buffer

Because the available bandwidth during the session fluctuates a lot, some mechanisms must be adopted to smooth the voice jitter. The basic idea of jitter buffer is to pre-buffer a certain number of slots of audio signal before start to play. Using the buffered data, the system can fill the lost slots caused by packet drop or bandwidth jitter. The advantage of this mechanism is that it does not need to synchronize the data. The jitter buffer can be dynamically set or deleted. But the jitter buffer introduces delay between the two end users. And there is trade off between the size of the jitter buffer and the real time data transfer.

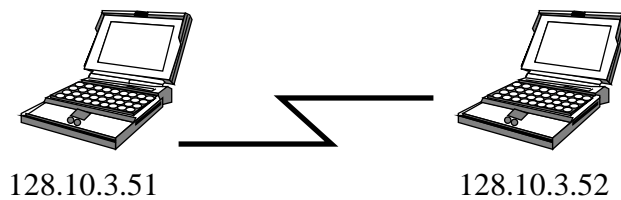
A double link list is implemented to manage the free buffer pool. A single link list is implemented to manage the received audio slots. Every slot carries a time or sequence tag generated by the source end. This tag is used to schedule the play order when the packets are out of order. Two pointers that point to the latest frame and oldest frame of received data separately are maintained. When a frame is received and the free buffer pool is empty, the oldest slot will be released and filled with the new data. It shows that we prefer the new data instead of the old one. A semaphore is established to achieve the exclusive operations on the jitter buffer.

Main functions of jitter buffer include:

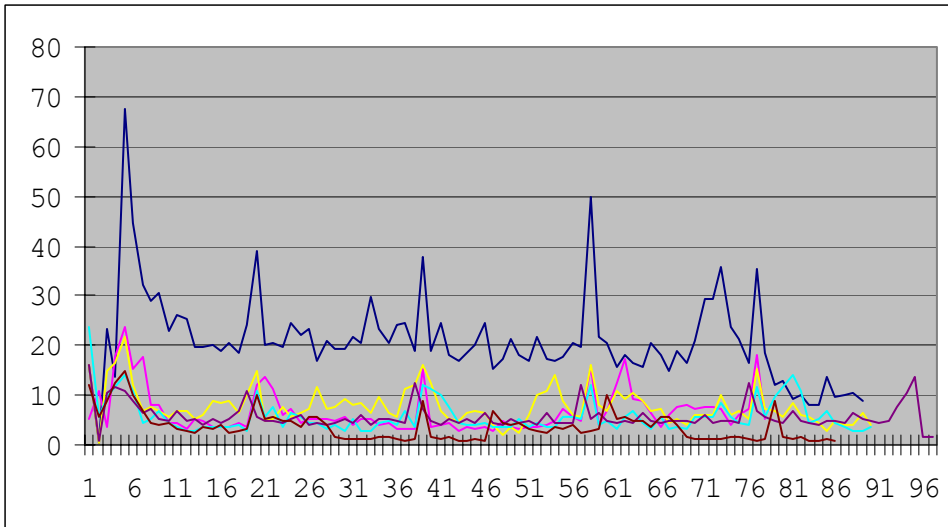
- JitterBufferRun():
- cooperate with network
 - QueueReadFrame()
 - GetReadFrame():
- cooperate with play back device
 - FreeWriteFrame()
 - GetWriteFrame(timestamp)

8 Part of Results

Experiment environment is established as follows. The sockets we use are 4000—4003 and 5000—5003.



The following figure shows some of the experiment results. The audio are all based on G723.1. The video are based on H.263. We adjust the ratio of inter and intra frames to examine the bandwidth usage. The frame rate is 3 frames per second. The bandwidth usage are examined every 300 ms. We collect the average and maximum bandwidth used by video codec. The examined inter/intra ratios are 5, 10, 15, 20, and 30.



Bandwidth usage: X-axis is time, Y-axis is consumed bandwidth

The table shows the maximum and average bandwidth consumption.

Divider Value	Avg	Max
Inter : Intra = 5	•21.4kb	•71.02kb
Inter : Intra = 10	•17.6kb	•47.45kb
Inter : Intra = 15	•13.08kb	•40.01kb
Inter : Intra = 20	•11.8kb	•27.35kb
Inter : Intra = 30	•6.8kb	•19.74kb
Sound Signal(G723.1)	•5.8 kb	•7.68kb

9 Future work

Several extensions can be added onto the current system. First a prototype for one-direction multimedia data transfer model will be established. With that model, mechanisms on multimedia database can be examined on this system. Secondly, other types of audio compression algorithms should be implemented and added into the system. Third, synchronization mechanisms that can map the audio and video signals should be implemented. The prototype is implemented on the Windows platform. It can be easily ported to Pocket PC system and examine the real time applications on mobile networks.

Class structure design

