VoIP
Technical Report

A technical report documenting a Voice Over IP system which allows communication between two computers on a network.

Matthew Williams : 100084873
Jake McVey : 100097848

Table of Contents

1. Network Analysis 2
1.1 Datagram Socket 2
1.2 Datagram Socket 2 2
1.3 Datagram Socket 3 2
1.4 Datagram Socket 4 2
2. Design of VoIP System 2
2.1 Datagram Socket 3
2.2 Datagram Socket 2 3
2.3 Datagram Socket 3 3
2.4 Datagram Socket 4 4
3. Evaluation of Final System 4
3.1 Decisions and Tradeoffs 4
3.2 Quality of Service 4
3.2.1 Bit Rate 5
3.2.2 Packet Efficiency 5
3.2.3 Speech Quality 5
3.2.4 Delay 5

1. Network Analysis

In order to analyze the four datagram sockets, 10,000 packets of data will be sent using each socket. The received data can then be compared to the sent data, and used to determine the characteristics of each socket. The most notable characteristics being packet loss, in-order arrival, and packet integrity.

1.1 Datagram Socket

When using `Datagram Socket 1` there was 0% packet loss, all packets arrived in the order in which they were sent, and the packets maintained integrity. These are the ideal characteristics for sending VoIP.

1.2 Datagram Socket 2

When using `Datagram Socket 2` there was an estimated packet loss of around 24.8%. This left noticeable gaps in the sent data, especially when bursts of loss occurred. Packets did arrive in order and maintained integrity.

1.3 Datagram Socket 3

When using `Datagram Socket 3` there was an estimated packet loss of around 16.4%, which again left large gaps in the data. Packets also arrived out of order, however integrity was maintained.

1.4 Datagram Socket 4

When using `Datagram Socket 4` there was 0% packet loss, and all packets arrived in order. However, packet integrity was compromised, with some of the packets being corrupted.

Table 01 - Datagram Socket Summary

Socket	Packet Loss	Correct Order?	Additional Notes
Datagram Socket	No - 0%	Yes	No problems.
Datagram Socket 2	Yes - ~24.8%	Yes	Frequent large bursts of loss.
Datagram Socket 3	Yes - ~16.4%	No	Packets do not arrive in order.
Datagram Socket 4	No - 0%	Yes	Packet integrity compromised.

2. Design of VoIP System

Each `Datagram Socket` required a different approach when attempting to improve the speech quality. Each frame of audio recorded was recorded and stored into a `byte array`. The recorded audio is then paired with a header and sent using the socket. A `byte array` is created on the receiver side in order to store the arriving packet. The receiver then removes the header and plays the audio in the best way possible.

2.1 Datagram Socket

After testing the characteristics of `Datagram 1` it was shown to be the “perfect” conditions for a VoIP system. Therefore, it was possible to increase the amount of frames stored in each packet, without consideration for faults such as packet loss. The sender side of the network now stores 3 frames of audio in each packet, this increases the overall packet size greatly but is more efficient.

2.2 Datagram Socket 2

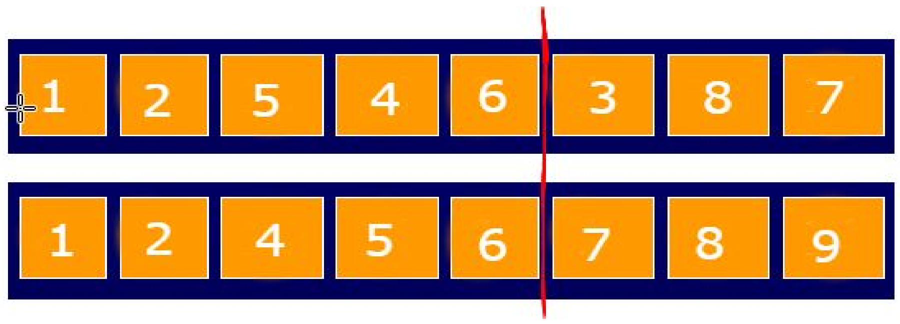
`Datagram 2` was found to have a packet loss of around 24.8%, which means it is important to know which packets are not arriving. The solution to this was to add a header to the packet with an `integer` representing the order in which the packet was sent. This could then be decoded on the other side to make it simple to identify which packets were lost.

Interleaving was implemented to reduce the effect of long bursts of packet loss. This worked well in terms of audio quality but added more delay. Repeating was used to fill gaps of lost packets by subtracting the `integer` stored in the header of the last successfully received packet from the `integer` in the header of the current packet.

2.3 Datagram Socket 3

`Datagram 3` experienced packet loss, at roughly 16.4%. Packets sent using this datagram also arrived out of order. Storing an integer in the header representing the order the packet was sent allowed the system to identify which packets were lost, as well as which were out of order. The next problem was to implement a way to sort the packets into the correct order. To do this, 5 packets were loaded into a buffer, and using the `integer` from the header, sorted into the correct order. This worked well for minimising the delay introduced, but did result in packets more than 5 positions out of order remained unsorted. To solve this, the system will ignore all out of order packets after sorting. As seen in *Figure 01*, since packet 3 is not within the first buffer 5, it is not sorted into the correct position, and is subsequently ignored and never played. To fill the gap, packet 2 will we repeated.

Figure 01 - Packet Group Sorting



An example of packets grouped into blocks of 5 and sorted.

2.4 Datagram Socket 4

`Datagram 4` had the problem of packet integrity being compromised before they arrived. The solution to this was to use cyclic redundancy checking (CRC). On the sender side a CRC method is used to obtain a `long` value based on the contents of the frame of voice data in the `byte array`. This value is then transmitted in the header, to be used to check for packet integrity on the receiver side. When the packet is received the same CRC method is called on the data, and if a different value is acquired, then a faulty packet has arrived. In this case, the system ignores the broken packet, and repeats the last packet that was received successfully.

3. Evaluation of Final System

Evaluating the overall quality of the system was done by running the same audio through each `datagram` and measuring the audio quality using PESQ. Other methods of evaluating such as calculating Bit Rate, Packet Efficiency, Delay, and overall Speech Quality were considered.

3.1 Decisions and Tradeoffs

Interleaving and repeating were found to be the most effective compensation methods, producing the highest quality audio overall. The problem with interleaving was the delay it caused. This was solved by using a smaller size. A 3x3 interleaver appeared to work the best in terms of quality / delay. Splicing the packets together if there was packet loss was something that was considered early on, however the audio quality was poor, which meant it was never considered again.

In `Datagram 1` it was decided that the system should send more than 1 block of audio per packet, since the network was known to have perfect conditions. This worked well and improved the efficiency as less sending occurred. This method was attempted with the other `Datagrams`, however it resulted in bigger blocks of missed data if a packet was corrupted or lost.

3.2 Quality of Service

Table 02 - PESQ Analysis

`Datagram 1` as expected received a perfect score. The reason for `Datagram 2’s` low score was the ineffectiveness of the interleaver implemented in the system. `Datagram 3 and 4` both received similar scores as the audio quality was acceptable.

Datagram Socket	**PESQ Score**
Datagram Socket 1	4.5000
Datagram Socket 2	2.5409
Datagram Socket 3	3.0979
Datagram Socket 4	3.0951

3.2.1 Bit Rate

Table 03 - Bit Rate Analysis

Datagram Socket	Packets Per Second	Packet Size (Bits)	Bit Rate (kbps)
Datagram Socket 1	31.25	12,288	384
Datagram Socket 2/3	31.25	4,128	129
Datagram Socket 4	31.25	4,192	131

3.2.2 Packet Efficiency

Table 04 - Packet Efficiency Analysis

Efficiency is measured by comparing the payload of a packet relative to the non-payload bytes.

Datagram Socket	Size of Information (Bytes)	Total Size of Packet (Bytes)	Efficiency
Datagram Socket 1	1536	1536	100%
Datagram Socket 2/3	512	516	99.22%
Datagram Socket 4	512	524	97.81%

3.2.3 Speech Quality

Table 05 - Speech Quality Analysis

10 Students were played the same audio over each datagram and asked to rate the quality.

Datagram Socket	5 Rating	4 Rating	3 Rating	2 Rating	1 Rating	Average Rating
Datagram Socket 1	8	2	-	-	-	4.8
Datagram Socket 2	-	-	6	4	-	2.6
Datagram Socket 3	-	4	5	1	-	3.3
Datagram Socket 4	1	6	3	-	-	3.8

3.2.4 Delay

Table 06 - Delay Analysis

The delay was acquired by timing the offset after 50 numbers were spoken into each datagram.

Datagram Socket	Delay (Seconds)	Reason for Delay
Datagram Socket 1	~0.3s	Tiny delay as expected.
Datagram Socket 2	~2.5s	Overhead from interleaving.
Datagram Socket 3	~1s	Loading packets into buffer before playing.
Datagram Socket 4	~0.3s	Tiny delay, likely from the CRC methods.