Networks Report

Alex Newark (100213127), Jack Hilsdon (100133297), Christopher Holder (100212682)

CONTRIBUTION %: 33.3% EACH

References

[1] *What is packet? - Definition from WhatIs.com. (2019). Retrieved from* https://searchnetworking.techtarget.com/definition/packet

[2] *What is packet loss? - Definition from WhatIs.com. (2019). Retrieved from https://searchnetworking.techtarget.com/definition/packet-loss*

[3]*Milner, B. (2019). Lecture 7: Transport and Network Layers. Presentation, University of East Anglia.*

[4] *Perkins, C., Hodson, O., & Hardman, V. (1998). A Survey of Packet-Loss Recovery Techniques for Streaming Audio. Network, IEEE, 12(40).*

1.0 Network Analysis

1.1 Datagram Socket 1

Figure 1 shows that there is no delay (running on localhost) and additionally no packet lost as the two graphs are perfectly mirrored. Furthermore when a header is added to the packet of Datagram socket 1 containing the packet id, the packets are received in the correct order when logged to console.

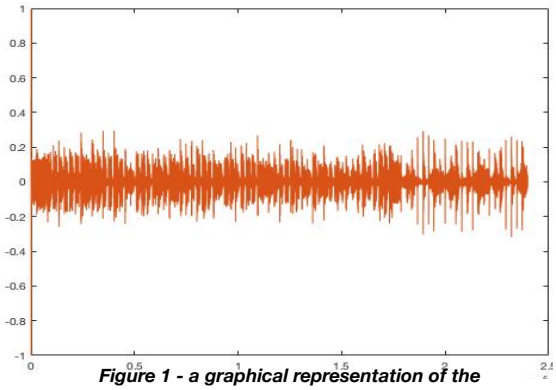


Figure 1 - a graphical representation of the output audio (orange) plotted on top of the input (blue)

1.2 Datagram Socket 2

As is demonstrated in figure 3 there is around a 23% packet loss (over 30 seconds) which occurs in medium and large bursts of loss which can be seen with some thin blue bars (such as at the start 0.1) and some relatively larger ones (such as at the end at 2.4). Additionally the loss in packets is spread apart inconsistently. For example between 1 and 1.25 in figure 2 there is a gap with no packet loss. However, between 0.75 and 1 there is a large amount of loss. The general 'bursts' Can be seen in the graph with some packet loss followed by no loss. Additionally figure 2 demonstrates a very slight delay in receiving packets (as can be seen the orange (output) occurring slightly after the blue (input)). Finally the audio appears to be amplified somewhat as the can be seen the orange bars in some instances (such as at 1 in the middle) are amplified version of blue.

	Data
Number Sent	912
Number Received	699
Number Lost	213
Loss percentage	23.3552631578947

Figure 3 - datagram 2 statistics

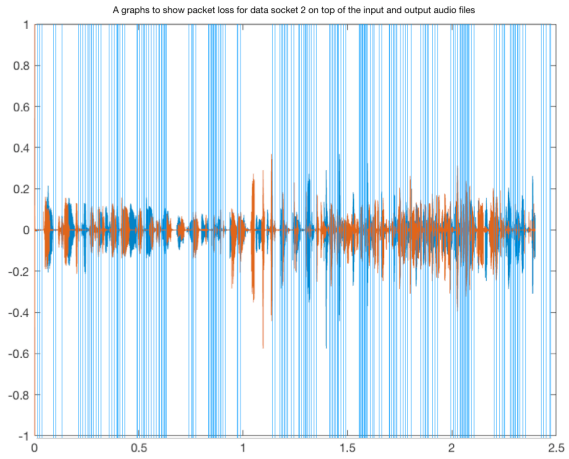


Figure 2 - a graphical representation of the output audio (orange) plotted on top of the input (blue) with the addition of packet loss overlayed on top (a blue bar represents packet for datagram 2

1.3 Datagram Socket 3

As demonstrated in figure 5 there is around a 18% packet loss (over 30 seconds) which occurs at a consistent rate throughout. The consistent packet loss is evidenced in figure 4 as the yellow bars seems to be consistent throughout and there is no single big burst, it's all small bursts at consistent intervals. Additionally the packets do not arrive in the order sent. Figure 6 demonstrates the console log of the order the packets were received (the integer value represents the order they were sent in e.g. 1 was sent first, 2 sent second etc.). Additionally a there is a very slight delay when receiving the packets (can be seen as audio files not mirrored in figure 4.) Finally there is slight amplification of the sample when receiving. For example at 1.5 in figure 4 it appears the audio is being amplified slightly.

	Test1
Number Sent	896
Number Received	732
Number Lost	164
Loss percentage	18.3035714285714

Figure 5 - datagram 3 statistics

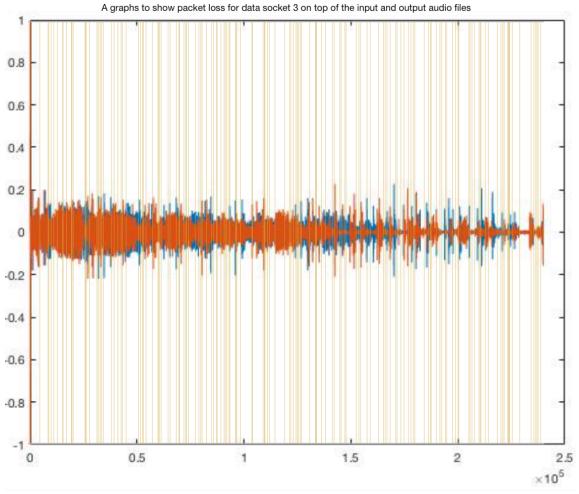


Figure 4 - a graphical representation of the output audio (orange) plotted on top of the input (blue) with the addition of packet loss overlayed on top (a blue bar represents packet for datagram 3

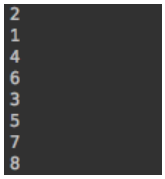


Figure 6 - datagram 3 packet received order in console

1.4 Datagram Socket 4

Figure 7 demonstrates that packets over datagram socket 4 are being corrupted. On average as figure 7 shows 10.1% (on average) are being corrupted. This data was collected by counting all packets received (to get the total received) followed by the total number that made it through the security protocols. The data corrupted in the packet is the audio body (the first 512 bytes) which can be seen both because the security protocols (discussed in section 3.0) detect a change but also when running over datagram with not security filter, and listening a distinct background tick noise can be heard suggesting corruption.

Iteration	Total sent	Total received	Total received not corrupted	Total corrupted packets	Percentage
1	924	924	834	90	9.74025974025974
2	917	917	822	95	10.3598691384951
3	925	925	835	90	9.72972972972973
4	924	924	821	103	11.1471861471861
5	926	926	837	89	9.61123110151188
Averages	923.2	923.2	829.8	93.4	10.1176551714365

Figure 7 - Packet received and corrupt table for datagram socket 4

2.0 Error mitigation

2.0 Overview

When data is passed across a network, it is done through 'packets'. "A packet is the unit of data that is routed between an origin and a destination on the Internet or any other packet-switched network"[1]. This data while traveling across a network is subject to 'packet loss' which is the "the failure of one or more transmitted packets to arrive at their destination"[2]. As a programmer of an application that interacts over a network (especially connecting to a wide-area network (WAN) with hundreds of different types of hardware and routers connected), developers have very little control over packet loss over the network and therefore have to implement error mitigation techniques (mitigation suggesting not stopping the loss but reducing the losses impact on the product). In this report these mitigation techniques will be split into two distinct sections: receiver based mitigation and sender based mitigation. The name implies where the mitigation technique will be performed i.e. when the packet is sent, or when it is received.

2.1 Receiver based mitigation

Receiver mitigation is any error mitigation that is performed where packets are received.

2.1.1 Splicing

Splicing is the a mitigation technique that focuses on filtering out gaps in audio transmission (cause by packet loss) by 'gluing together the next available frame. This improves intelligibility by allowing the audio to continuously flow.

Figure 8 shows what happens when splicing occurs. The lost packet is represented by the red line which is removed by moving the two audio waves together and therefore removing the gap.

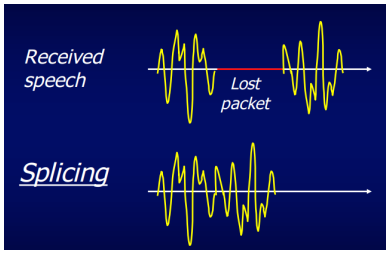


Figure 8 - Splicing an audio wave.[3]

While one could argue when it comes to a system such as Voice Over IP (VoIP) having 'some' audio replace missing packets could be beneficial as it's better to have something than nothing, it will be demonstrated in section 4.0 this is not the case and for the context of our datagrams sockets especially poor as it has been found that when implementing splicing, packet "loses above 3 percent"[4] are "intolerable"[4] when using splicing.

Splicing has low computation complexity and therefore is good for a VoIP system as this reduces the delay.

Additionally when it comes to a VoIP system that uses audio the timing at which an audio file is played is of significant importance. Splicing together two audio packets significantly impacts the jitter of the given audio packet as it reduces the overall playback time of the audio.

Overall this implies splicing works well over short bursts of packet loss which are infrequent and under 3% of overall packet loss but not as well anything above 3% due to the negative affect on jitter and quality.

2.1.2 Fill-in

Fill-in is using noise or silence in place of a lost packet. This is a way of covering up gaps in audio, whilst maintaining the length of playback.

Figure 9 demonstrates adding in noise into a los packet area. What is of important note is the audio length does not change and playback time will remain the same (compared to figure 8 where audio spliced reduces the length).

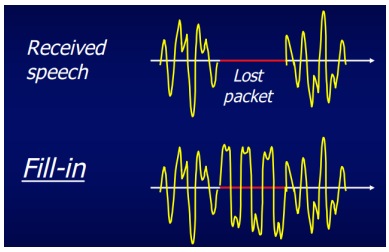


Figure 9 - Fill-in an audio wave.[3]

While fill-in overcomes the jitter issue with splicing, it does however also only work at low amounts of packet lost and especially only over small bursts. If for example you have a large burst of packet loss

and you're inserting nothing, there will only be silence until a packet that isn't loss comes. This is very inconvenient and not ideal of a VoIP system.
Overall fill-in could be considered better than splicing as it overcomes the jitter issues that face splicing however, it is very limited as well because it can only be used at short bursts of packet loss and not much packet loss overall.

2.1.3 Repetition

Repetition is the process of repeating the last available packet to fill in the gaps left due to packet loss.
Figure 10 shows the process of taking the previous packet received and inserting it into the area of packet loss.
Similarly to fill-in this means that the length of the audio does not change and Therefore jitter is reduced. Furthermore, under small bursts of packet lost due to the way human perceive sound this means that one or two packets with repetition will not be noticed.

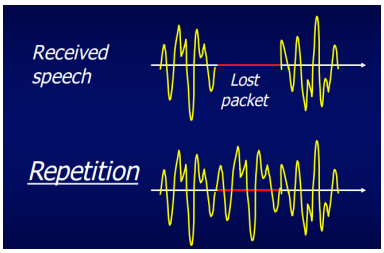


Figure 10 - Repetition of audio wave.[3]

However, if there are large bursts of packet loss and the same sound sample is inserted over and over again it will sound like the VoIP system has got stuck, and is very noticeable.
Another significant advantage of repetition is the low computation complexity which is an important factor to consider for a VoIP system as keeping computational complexity low will reduce the delay and therefore improve the quality of service.
Overall repetition is probably better than fill-in as the inserted packet sounds better as the sound of the previous packet (due to the way humans speak and perceive sound) will sound like it fits. However, this is quickly exposed if there are larger bursts of packet loss.

2.1.4 Interpolation

Interpolation is similar to repetition however, it takes the first half of the packet before it, and half of the first packet that follows (after the packet loss) and creates a new packet of the two halves, that is inserted into where the missing packet is. (demonstrated in figure 11).
Similarly due to the way humans perceive sound interpolation is very good at concealing a lost packet. It is slightly better at doing this than repetition as the sound is different and therefore can survive without being noticed over larger bursts of packet loss than repetition can.

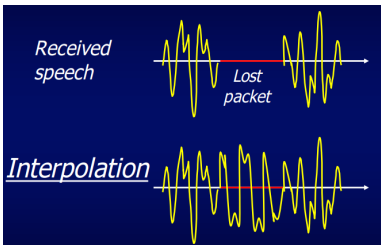


Figure 11- Interpolation of audio wave.[3]

However, interpolation has a high computational complexity due to having to search byte arrays and additionally having to wait for the next packet to arrive before it can be applied. This is very detrimental to a VoIP systems as the delay needs to be reduced as much as possible for the best quality of service.

2.2 Sender based mitigation

2.2.1 Interleaving

Many of the error mitigation and concealment mentioned above work well over small bursts of packet loss however, not so well over larger bursts of packet loss. An inteleaver reduces the chance of large bursts of missing packets occurring by changing the order in which the packets are sent out and therefore reducing the chance that there is a large burst of packet loss.
Packets are given a respective sequence number which is incorporated into the header and are passed through an interleaver block. This block is then rotated 90^0, dispersing the order of the packets for transmission. Once transmitted, the order is the reassembled using its' sequence on the receiver side.
The benefit in doing this is once transmitted, the packets that are lost won't be close together in the sequence, so the errors are dispersed which allows for a higher quality of service from our receiver-based mitigation therefore improving the intelligibility of speech.

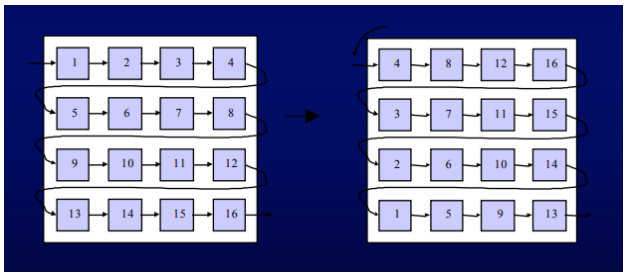


Figure 12- DxD Block Interleaver .[3]

Interleaving is not without its' downsides, however. There is a trade-off between the size of the interleaver and the delay having an interleaver causes. As the size of the interleaving block increases, as does the delay due to having to wait for DxD packets to be assembled, transposed and then rotated.
This has significant computation complexity cost and for a VoIP while an interleaved is exceptional at reducing the change of large bursts of packet loss occurs, improving the quality of service the delay may reduce the QoS an equal amount. Therefore a balance must be struct to find the optimal size of interleaver.

3.0 Security

When developing the VoIP system's security layer, there's a lot that had to be considered. There are different ways that the system could be attacked, and so it should be defended from as many angles as possible.

The security layer was designed to be replaceable. This means that if any part of the security is breached in future, a new layer can easily be developed and put in place. It also made it easy to test the VoIP system without any encryption or decryption, which is beneficial to development, testing and maintenance. To accomplish this, an interface was used, ISecurityLayer, with encrypt and decrypt methods. The current implementation of this is known as BasicSecurityLayer.

3.1 Message Authentication

The first decision that had to be made was how to authenticate users, and their messages. To be able to securely encrypt and decrypt packets, system users need to have a shared, confidential understanding of packet data, an easy way to do this is to encrypt using a private key.

There are many ways to implement a private key, however with a Java binary reverse engineering is a great concern. Because of this, a compile-time key is unsuitable, and instead a private key is entered by the user at runtime. A runtime key has other benefits as well, if it is compromised, the system can be rebooted, with a new key entered at startup to ensure long-term privacy. This is not possible with a compile-time key, as a rebuild would be required to change the key.

Now that the users have a secret, a trusting mechanism can be implemented. A very basic approach would be to share a message authentication key with the packet, however this is very insecure. Anyone who obtains a packet can identify the key and begin using it in an attack. A better method is to create a checksum of the audio data before encryption, and send it. Then, the receiver can decrypt the audio, calculate the checksum, and compare it to what was received. The VoIP system uses a SHA-512 checksum, and any invalid packets are discarded. Another easy way to identify bad packets is to simply check their length. Packets which are too long or short are also discarded. With both of these checks in place, receivers can only hear senders using the same system and private key.

3.2 Message Confidentiality

When sending sensitive information across a network, it's important to ensure it's confidentiality. It is possible that a third party could intercept packets of data, and so it's important that they aren't able to understand it. To audio data is encrypted, using a method only known to the sender and receiver. This VoIP system uses two methods of encryption: a shift cipher and an xor cipher.

3.2.1 Shift Cipher

A shift cipher is a basic cipher that involves shifting data to an offset position within an array. When the end of the array is reached, data wraps back around to the other side.

For example, when shifting an array of bits 117 bits, the audio data will begin at array position 117, and end at 116. Figure 13 demonstrates a how a shift cipher is working.

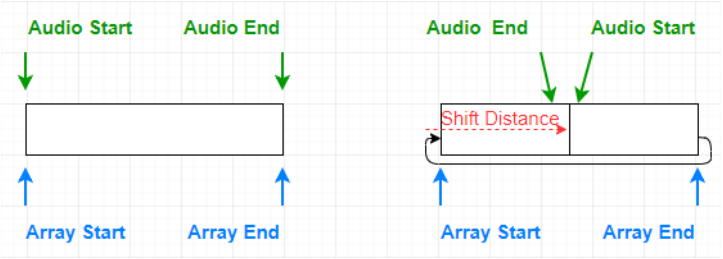


Figure 13- Shift cipher

Java doesn't provide many options for manipulating bits, and works mostly on multiples of bytes. So BasicSecurityLayer pushes the array in intervals of bytes, specifically 117 bytes. This value is hardcoded, as higher values may impact performance, and the private key can reach the integer limit. This does mean that the value could be discovered through decompilation, but it also means that it can be controlled by the program builder, replaced and updated in future. Additionally, it means that the if the private key is compromised, it's likely that the shift cipher will still hold up.

3.2.2 XOR Cipher

Another basic cipher used by BasicSecurityLayer is an XOR cipher. An XOR (exclusive-or) operation is performed with the two operands being the integer private key, and an integer of audio data. Reversing the cipher is simple, another XOR operation with operands of the private key and encrypted data yields the original audio data.

Audio Data	0	1	1	0	1	0	1	1
Private Key	0	0	1	1	0	1	1	0
Encryption Result	0	1	0	1	1	1	0	1

Figure 14- XOR Cipher

4.0 Evaluation of VoIP system in terms of QoS parameters

4.1 VoIP QoS overview

4.1.1 VoIP considerations

When designing a VoIP system three main characteristics define the quality of service (QoS):
1) Average end-to-end delay, 2) Delay jitter and 3) Packet delivery ratio (or quality of sound received). Each of these will be evaluated with consideration of the different parameters that can be applied to the VoIP system for each socket.

4.1.2 VoIP parameters

The VoIP system implements five parameters to consider when assessing QoS: 1) Splicing, 2) Interpolation, 3) Repetition, 4) Interleaver, and 5) Size of Interleaver 'blocks'

4.2 Parameter evaluation

Each parameter will be evaluated in terms of the VoIP QoS considerations and applied to a specific datagram socket to provide evidence of the argument.

4.2.1 Bitrate

The bitrate per sample for the VoIP 16 bits (2 bytes) which had a sample 256 per packet (512 bytes). As you increase the bitrate of the audio sending in theory the overall quality should go up. However, a higher bitrate takes up more bandwidth which some do not have enough of to use. Therefore when setting the bitrates it's important to consider who and what kind of hardware and network will they have as if they don't have the bandwidth to support the bitrate of the VoIP system then the increasing the bitrate will be more detrimental than beneficial. Therefore, the overall benefit to QoS is subject to if the user can support the increased bandwidth.

4.2.2 Interleaving

As discussed in section 2.2.1 interleaving goal is to reduce large bursts of packet loss by reordering the order in which packets are sent and additionally being able to cope with packets arriving in the wrong order. As shown in figure 4 and section 1.3 datagram socket 3 is subject to packet loss and the packets arriving in the wrong order. Therefore implementing an interleaver should greatly improve the results and quality of service in terms of packet delivery ration. For this example we will set the block size to 4 (will discuss size impact in 4.2.2).
With the interleaver implemented the average ms to receive a packet went from 0.135 seconds to 0.48 seconds (see figure 15). This is just under the the defined 0.5 seconds accepted delay and therefore the improvement reflected by the interleaved are positive for the VoIP system.
In terms of quality improve a significant average rise from 1.3899 PESQ score with no interleaver to 2.6343 is a very significant improvement to the quality of audio.
However, with the delay time rising so exponentially (and this is on localhost) even with the improved score the interleaver (currently unoptimised) may lead to a decrease in QoS if the size and optimisation of it into the future isn't well managed.
Finally in order to implement interleaving a header had to be added to each packet. Therefore focus was placed on packet efficiency when constructing these headers. The overall packet size was 526 and conformed to the standard RTP header fields. The fields and sizes were as follows: version (1 byte), padding (1 byte), extension (1 byte), csrc count (1 bytes), marker (1 byte), payloadType (1 byte), sequence number (1 byte), timeStamp (6 bytes), synchronization source and identifier (1 byte) . Therefore, the packet efficiency is 512/526*100 = 97.3%.

4.2.3 Interleaver block size

As discussed in section 4.2.1 while an interleaver saw great increases in PESQ score it does however,(as displayed in figure 15) lead to exponential delay as the interleaved size increases. The time taken to send and receive the packets increases exponentially (blue line) where as score increases at a logarithmic rate (green). This means that the interleavers trade off, of increasing delay leading to better score is quickly outweighed when considering QoS. Therefore, it is very important to choose carefully the size of the interleaver.

Size	PESQ score (Test 1)	PESQ score (Test 2)	PESQ score (Test 3)	Time (seconds)	Average
No interleaved	0.9305	1.9065	1.3329	0.135	1.38996666666667
2x2	2.6725	1.7412	1.7986	0.134	2.07076666666667
3x3	2.0945	1.8048	2.7254	0.387	2.20823333333333
4x4	2.4579	2.4306	3.0146	0.48	2.63436666666667
5x5	1.7583	2.2426	3.3344	0.894	2.4451
6x6	2.3944	2.1081	2.5813	1.161	2.36126666666667
7x7	2.0883	2.5115	2.5720	1.674	2.3906
8x8	2.1441	2.2789	2.55321	2.059	2.32540333333333

Figure 15- Interleaver score and time taken

The ideal delay for a VoIP system is at max 0.5 seconds. All sizes under 4x4 in figure 15 meet this criteria. Therefore, it may seem clear to choose a 4x4 however, the time is 0.48 seconds (figure 15) which is cutting it very close. In addition as will be shown in section 4.2.3,4.2.4 and 4.2.5 an interleaver is best used in conjunction with receiver compensation. Therefore when assessing the parameters below testing will be don't without a interleaver, with a 4x4 and 3x3 (any bigger would be to time consuming and not meet the QoS).

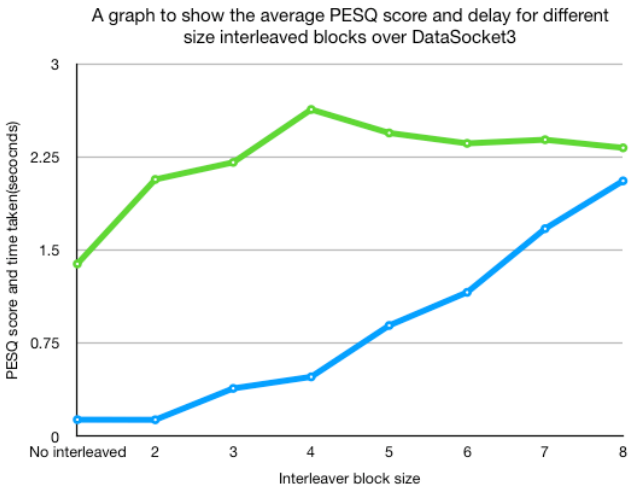
Figure 16- Interleaver score (green) and time taken graph (blue)

4.2.4 Receiver based compensation

Section 2.1.3 proposed receiver based compensation works best with short bursts of packet loss. With the implementation of the interleaver this reduces the chance of large bursts of packet loss and therefore in conjunction with an interleaver much better results for each compensation method should be seen. This will be demonstrated over datagram socket 2, because as shown in section 1.2 datagram socket 2 is exposed to large bursts of packet loss.

4.2.5 Repetition

Repetition as stated in section 2.1.3 has a low computation complexity and therefore when assessing the QoS hopefully the delay will be low delay will merit improved result on the base interleaver and on its own. As shown in figure 17 repetitions average score without interleaver was only 1.61 with a 0.134 second delay. However, this improved in a 4x4 interleaver with a average score of 2.28 and a 0.521 second delay.
While the overall improvement in QoS in terms of quality has greatly improve the delay becomes greater than 0.5 seconds with a 4x4 and therefore in the current state of the interleaver (unoptimised) the use of a 3x3 is potentially the best case. As not only does it maintain a very high scoring PESQ but its delay is 2.5. Overall repetition on its own is poor for QoS but in conjunction with an interleaver (3x3 is best) it is very effective at improving the QoS.



Receiver based compensation with interleaver

Techniques (interleaver size)	PESQ score (Test 1)	PESQ score (Test 2)	PESQ score (Test 3)	Average score	Time (seconds)
Reptition (non)	2.0210	1.4307	1.3818	1.61116666666667	0.134
Interpolation (non)	1.4512	2.1886	1.9137	1.85116666666667	0.138
Splicing (non)	1.3269	1.4572	2.2984	1.69416666666667	0.135
Reptition (3x3)	2.5506	2.5815	2.5180	2.55003333333333	0.257
Interpolation (3x3)	1.9618	2.4099	1.9016	2.0911	0.266
Splicing (3x3)	1.6367	2.4011	1.4580	1.83193333333333	0.262
Reptition (4x4)	2.2875	2.2900	2.2825	2.28666666666667	0.517
Interpolation (4x4)	2.0781	1.9409	2.1064	2.0418	0.521
Splicing (4x4)	2.8895	2.4357	2.5470	2.62406666666667	0.515

Figure 17- Interleaver effect on receiver compensation

4.2.6 Splicing

Splicing has very similar impacts on QoS as repetition however, is slightly more expensive in terms of computational complexity (Splicing alone 0.135 seconds in length to run alone and repetition is 0.134 seconds). However, over the course of the table figure 17 while improvements to QoS were seen repetition was better at improving the QoS across the
board. Additionally it appears that as the size of the of the interleaver matrix increases the splicing delay grows much faster than repetition.

4.2.7 Interpolation

Finally interpolation while it has a more expensive computational complexity than repetition and splicing, which on its own is worse for the QoS due to the increased delay, it appears to make up for this if used in smaller interleaver matrix size. On its own and with a interleaver at lower interleaver sizes, it makes up for the delay on it's of 0.138 by being able to gain higher PESQ scores with a small interleaver or no interleaver at all. This is important to take into account because if we need to achieve the highest QoS with the smallest delay an interleaver under a 3x3 and interpolation appear to be most effective.

Final VoIP implementation techniques for each socket

Datagram socket 1 : The final solution didn't implement any of the compensation techniques mentioned. This is because it had no packet loss and therefore adding anything extra would be unnecessary and decrease the QoS for the VoIP by increasing delay unnecessarily.
Datagram socket 2: The final solution implemented an interleaver of size 3x3 (ideally would like to use 4x4 however, interleaver code not optimised enough) with repetition. This saw a delay of 0.25 seconds and an. Average PESQ score of 2.55. This was the best result when considering the QoS for a VoIP system. The most important technique Implemented was the interleaver due to the large amounts of inconsistent packet burst which without the interleaver saw very detrimental results of under 1.3 PESQ score.
Datagram socket 3: The final solution implemented for data socket 3 was an interleaver of size 4x4 and interpolation. While this combination with the size of the matrix being 4x4 lead to a delay greater than 0.5 seconds, due to the consistent loss of packets it was determined the ability to hear and have a conversation over the network outweighed the costs in delay (as without a 4x4 and interpolation it is nearly inaudible). The interleaver is the most important part of this due to packets coming in the wrong order which the interleaver made redundant.
Datagram socket 4: Packet data sent through socket 4 was often corrupted when received. The checksum implemented for security purposes was used to identify bad packet data, and corrupted packets were discarded. The gaps were interpolated