Topic 1: A Better UDP

Carson Clanton, *Member, Phi Mu Epsilon,* Kevin Hayes, William Hunter, Richard Meth, and Ryan Nordeen

***Abstract*—This article describes an approach to improving the reliability of UDP. Our approach uses sequence numbers for packets to force in-order delivery to the Network Application layer and Error-Correcting Codes to help reproduce lost packets without requesting retransmission of these missing packets. The level of reproducibility will be largely dependent on the Error-Correcting Code chosen and the selection of its associcated redundancy parameters.**

***Index Terms*—UDP, Reliable, Error-Correcting Codes, Sequence Numbers.**

I. Introduction

WHEN a software engineer begins designing a network application, he or she has to make a decision between two transport layer options, TCP or UDP. There are advantages and disadvantages to both. With UDP comes nimbleness and low overhead, along with dropped and possibly out-of-sequence packets. With TCP comes reliable stream-based communication at the expense of throughput. There isn't much middle ground between the two. The approach discussed in this papar attempts to change that.

We don't attempt or intend to redesign either protocol. Instead, we seek to enhance the reliability of UDP, so that it is a little more like TCP, while remaining connectionless. To do this, we will require two added features to be built "on top" of UDP: Sequence Numbers and Error-Correcting Codes.

The introduction of sequence numbers should be somewhat obvious; on the receiving end of the UDP pathway there should be someway to put packets back in order. The subtlety, however, of requiring sequence numbers is that in order for our error-correcting codes approach to work, for the code we experimented with, at least 75% of the packet payload data must be available and properly sequenced. So, the inclusion of sequence numbers is twofold.

The second feature to be added, Error-Correcting Codes or ECC, is a little more complicated and requires more of an introduction, if for no other reason than to get acclimated to the terminology in the literature.

Given the no-guarantty and low-reliability characteristics of the networking layer and the little added value of UDP, a virtual communication link through a UDP socket represents an unreliable, and potentially noisy, channel, making it a good candidate for a ECC solution.

A. Error-Correcting Codes

Below in Figure 1 is a schematic of the general communication link for sending messages. Here the message is first sent by the *source* to the *encoder* where the message is assigned a *codeword*, i.e. a string of characters from some chosen alphabet. The encoded message is sent through a *channel*, which is certain to have some level of noise that may possibly alter the codeword. This possibly-altered codeword is then received by the *decoder* where it is matched with its most likely candidate codeword which is in turn translated into a message resembling the original message and passed on to the *receiver*. The degree of resemblance to the original message depends on how appropriate the code is in relation to the channel.

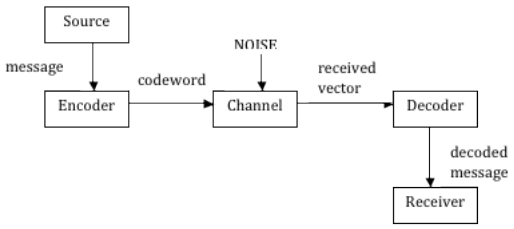

Fig. 1. Schematic of a general communication link

Example I.1. Suppose we are using an alphabet containing only two symbols, say 0 and 1 and we wish to send a message to a friend of either "YES" or "NO". We first might consider encoding 1 for "YES" and 0 for "NO"; however, if the codeword 0 is sent and then altered by the channel to be received as 1, the decoder will incorrectly pass the message YES to the receiver. One way of improving the situation is to add redundancy by repeating the symbol. For instance, we might encode "YES" as the codeword 11 and "NO" as 00. Now if NO is sent, two errors would have to occur before the decoder returns an incorrect message. If one error occurs then the received vector will either be 10 or 01, neither of which is a codeword. At this point, the receiver might request a retransmission. This is an example of a code in which one error may be detected. To strengthen the code to be error-correcting we might increase the redundancy by repeating the message five times. Thus we encode our message "NO" as 00000. Perhaps the channels interferes to cause the decoder to receive the vector 00110. Using the method of nearest neighbor decoding the decoder assesses the message and decides that of the two possible codewords (i.e. 00000 and 11111) 00000 is more likely the one originally sent and hence is correctly decoded as "NO".

1) Subsubsection Heading Here: Subsubsection text here.

II. Conclusion

The conclusion goes here.

APPENDIX A
PROOF OF THE FIRST ZONKLAR EQUATION

Appendix one text goes here.

APPENDIX B

Appendix two text goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.