Int. J. Communications, Network and System Sciences, 2025, 18(3), 27-38
https://www.scirp.org/journal/ijcns
ISSN Online: 1913-3723
ISSN Print: 1913-3715

TCP and UDP Control Bits Coded to Reduce Errors and/or Retransmissions

Chrislin Martial Lélé

GET, ENSP Yaoundé, Yaoundé, Cameroon
Email: chrislinlele@gmail.com

How to cite this paper: Lélé, C.M. (2025) TCP and UDP Control Bits Coded to Reduce Errors and/or Retransmissions. *Int. J. Communications, Network and System Sciences*, 1**8**, 27-38.
https://doi.org/10.4236/ijcns.2025.183003

Received: December 13, 2024
Accepted: March 28, 2025
Published: March 31, 2025

Abstract

This article is about TCP and UDP transport layer control bits. The aim is to encode some control bits in TCP and UDP headers in order to improve their reliability. This will ensure that some packets that arrive at the receiving side with only bit errors in the header part can be corrected and thus be delivered to the application layer without problem instead of being retransmitted or discarded. This is done by suggesting an adaptive scaling of the source port number and destination port number fields. In the case of TCP, this will provide in some cases, enough bits that will be added to the bits of the Urgent pointer field, Data-offset part, and the reserved field on a quest of a good coding rate.

Keywords

TCP, UDP, Reliability, Control Data, Coding

1. Introduction

As defined in the OSI (Open System Interconnection) model, the transport layer aims at providing communication between two processes on two different end-systems. Two main transport layer protocols are the subject of this paper: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Reliability often plays an important role in communication between processes. On one hand, in order to provide reliability TCP [1]-[5] has some mechanisms such as checksum and retransmissions. In such an approach the control information i.e. data of the header part of TCP segment are very important. The goal of this article is to use the structure of TCP header bits in order to encode the control data in such a way that its reliability improves and, therefore, the performance of the checksum mechanism. On the other hand, UDP [3] [5]-[7] doesn't provide reliability but the destination and source port numbers are critical parameters in order for the data

to be delivered to the right processes. Improving the reliability of source and destination port numbers will certainly improve the transmission process of the overall data. This article aims to provide an approach for improving transport layer communication. We also make use of the habits of users when they go to the Internet. For many users, there is a quite significant probability that they will not have more than 4 or 16 TCP or UDP connections open at the same time, therefore the number of bits assigned for port numbers appears to be high. Based on this observation we derive an algorithm for the mentioned purpose. This paper is divided in four sections. Section II presents UDP and the steps used to derive an algorithm to encode critical control information. Then, section III deals with TCP, its control information and the algorithm used to encode some control data. After that, the section Results and Discussion, gives an example of coding gain we can obtain with a convolutional code then some relevant issues from this result are detailed. At the end, a section entitled conclusion summarizes the main ideas and results of the article. Now, we start with the UDP protocol.

2. UDP

This section is about UDP. Firstly, we are going to present the control information in UDP header, then we give details of the algorithm that we are suggesting to improve reliability of some control information.

2.1. Control Information in UDP Header

The purpose of this sub-section is to sort UDP control information in two parts: critical (sensitive) and non-critical for the communication procedure. This part focuses on UDP header and the impact of undetected errors in the header part of its segment. Figure 1 illustrates the different fields which appear in a UDP packet. The first step is to discriminate between critical (sensitive) control information for successful communication and those that are not.

Sensitive control information:

Below is a list of UDP header fields that we consider to be sensitive information for transport layer transmission:

- Source port number: useful for sending back information.
- Destination port number: useful for selecting the appropriate process to which the data will be delivered.
- Checksum: used for error detection.

Let us describe briefly the importance of these critical parameters. We start with the Destination port number. The destination port number is a critical parameter because an error in this field will result in the data either being delivered to the wrong process or the process not being able to exist. The process doesn't exist in other words, it is as if the data is lost. The fact that the data can be delivered to a wrong process makes this field a very important one to protect. Concerning the source port number, it is also important as it corresponds to the sender's address, which is useful when sending back replies. Finally, the Checksum field is for im-

portant usage because it helps to perform the error detection functionality of the UDP protocol. Therefore, we can imagine that improving its reliability will improve the performance of the error-detection step. What can we say about non-sensitive control information?

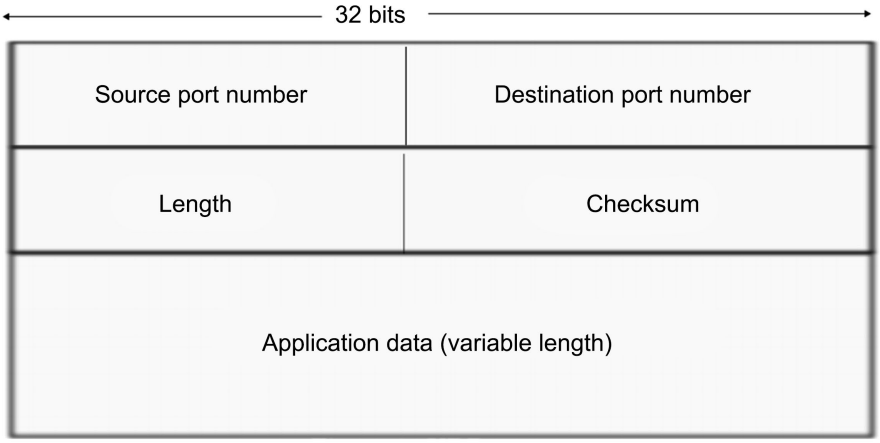

Figure 1. UDP segment.

Non-sensitive control information of UDP:

In this sub-section we are going to talk about UDP header fields that we consider to be non-sensitive with regard to communication between processes while explaining the reason of our choice. The field that we consider as non-sensitive is the Length field. The Length field data determines the length of the segment but if we use a principle similar to TCP by setting a *MSS* (Maximum Segment Size) then this field becomes useless and therefore it can be used for encoding. In this paper, we are making the suggestion to set a maximum size for UDP and then used the Length field for encoding. If fact, if the stream of bits to be sent is of size M bits, then we will have $n = \lfloor M/MSS \rfloor$ segments ($\lfloor y \rfloor$ represents the integer part of y) and the last segment will have a size equal to $n1 = M - n \times MSS < MSS$ bits therefore we propose using 2 bits (for example the first two bits) of the Length field to distinguish between the n packets of size *MSS* and the only packet of size $n1 < MSS$. This last packet can be handled by using the remaining bits of the Length field to specify its length. This means that this last packet will be processed as what normally happens in UDP. As the two first bits are concerned an example of possible assignment is given by:

- 00 meaning that the size of the data field is *MSS.*
- 11 meaning that the size of the data field is less than *MSS.*

In this paper, we will not care about the case where the size is less than *MSS* since the manner of dealing with that is straightforward. We plan to use this field for coding and we do not plan to protect non-sensitive control information with a coding. However, one must derive from this work an algorithm that also includes this field. Table 1 summarizes the classification and assumption we are using for the rest of the paper. In this table "not-used" means that we will use this

field for encoding. Having identified the critical fields in the header part of UDP, the next step is to improve their reliability.

Table 1. Summary of the classification and assumption.

Sensitive control data	Non-sensitive control data
• Source port number • Destination port number • Checksum	Length (not-used)
Size of the UDP header 8 bytes	

2.2. Algorithm for Coding Some Control Data of UDP

This sub-section deals with the algorithm that we propose in order to encode source and destination port numbers as well as the checksum. We assume that a principle similar to *MSS* in TCP is used, therefore the Length field will be used for encoding process. Moreover by analysis users' habits, we realized that a great portion of users will not have at the same time more than 4 or 16 TCP and UDP sockets open at the same time. That is why we suggest re-scaling the bits used for source and destination port numbers in an adaptive way. As an illustration, we choose to use 4 states:

- First state: both parts (end-systems) of the communication have less than 4 TCP and UDP sockets opened. In this case, only 2 bits are required for a given port number instead of 16 bits.
- Second state: both parts (end-systems) of the communication have less than 16 TCP and UDP sockets opened and more than 4. In this case, only 4 bits is required for a given port number instead of 16 bits.
- Third state: both parts (end-systems) of the communication have less than 256 TCP and UDP sockets opened and more than 16. In this case, only 8 bits is required for a given port number instead of 16 bits.
- Fourth state: both parts (end-systems) of the communication have less than 65536 TCP and UDP sockets opened and more than 256. Here, all the bits are used for the port number.

In order to distinguish these four states, we need 2 bits, which can be taken from the Length field (let us used the two last bits). An example is given in Table 2:

Table 2. Example of assignment between state and bits.

State	Bits of the state
State 1	00
State 2	01
State 3	10
State 4	11

However, since states are linked to the fact that 00 are the two first bits of the Length field, we suggest combining bits of states and bits to distinguish size (either less or equal than *MSS*) by 5 Cases where we use 6 bits (of the Length field) to discriminate between then as described by Table 3. The assignment of bits is done in order for any two sequences of 6 bits to differ by at least 3 bits. That is, the Hamming distance is at least 3. This will add robustness on the transmitted sequence since 3 bits difference provides a self-ability to correct 1 bit error at the receiving side. It means that 10 bits will remain for the encoding procedure. The six bits will be sent by both parts of the communication to reflect their current state in a similar manner as information is carried by the window field in TCP header for flow control. Any change in one part will be reported to the other part so that both sides move to the same state throughout the communication. The part with the higher number of opened sockets determines the state of the communication.

Table 3. Example of assignment between size-state and bits.

6 bits used	purpose
000101	The size of the data field is MSS and the state is 1
001011	The size of the data field is MSS and the state is 2
011100	The size of the data field is MSS and the state is 3
110001	The size of the data field is MSS and the state is 4
111111	The size of the data field is less than *MSS*

When a state i is used it leaves a given amount of bits (as the number of bits of the port number is concerned) that are not used and this amount of bits adds to the remaining 10 bits of the Length field to give the total number of bits which are available for the encoding procedure. Table 4 gives the details of what we obtain from each state while providing the coding rate of each state. Remember that the data to be encoded are the port numbers and the checksum.

Table 4. UDP: states and coding rate.

States	Length of control data to encode	Length of header field	Coding rate
1	4 + 16	64	20/58
2	8 + 16	64	24/58
3	16 + 16	64	32/58
4	32 + 16	64	48/58

Then an appropriate coding scheme such convolutional codes, turbo codes, linear block codes [8] [9] can be used. It is worth mentioning that the scaling of the

port numbers in states can be modified in many ways. That is instead of 4 states we can decide to have 8 states and so forth. After presenting the UDP protocol and our method to improve the reliability of its critical control data, let us focus on TCP.

3. TCP

This section is about TCP. We starts by presenting the control information in TCP header, after that, we give details of the algorithm that we are suggesting to improve reliability of some control information

3.1. Control Information in TCP Header

This sub-section aims to sort TCP control information in two parts: critical (sensitive) and non-critical for reliability procedure. It focuses on TCP header and the impact of undetected errors in the header part of its segment. Figure 2 illustrates the different fields which appear in a TCP packet. As in the previous section, the first step is to discriminate between critical (sensitive) control information for reliable process and those which are not.

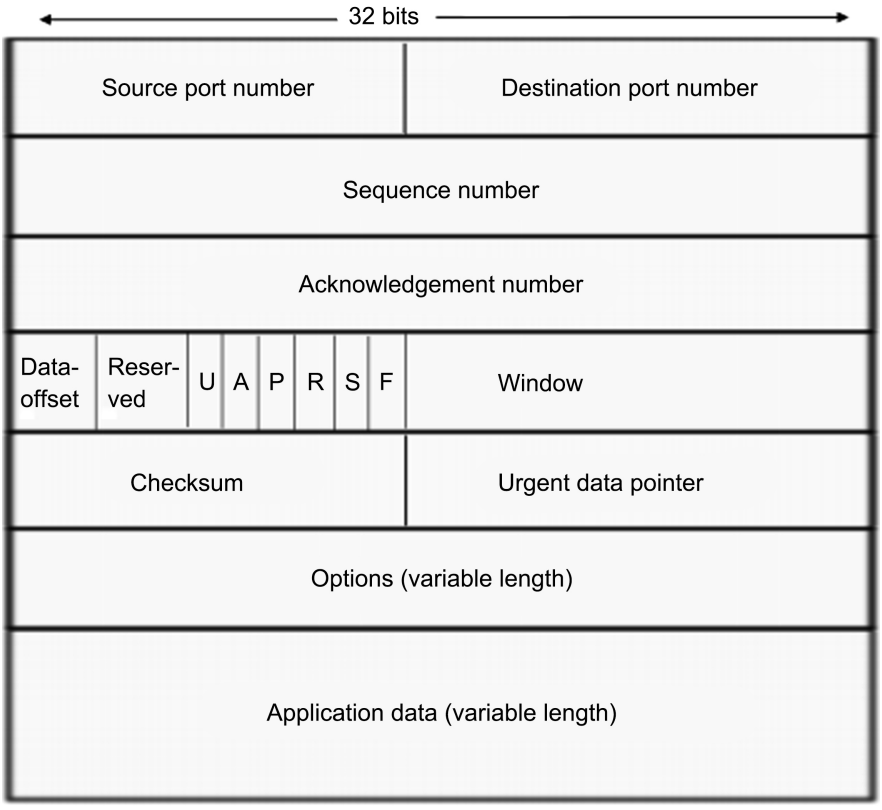

Figure 2. TCP segment.

Sensitive control information:

Below is a list of TCP header fields that we consider to be sensitive information for reliable procedure:

- Source port number: useful for sending back information.
- Destination port number: useful to select the appropriate process to which the data will be delivered.
- Sequence number: important field as it enables to put the flux of information in order.
- RST, SYN and FIN: important as they are used for connection setup or teardown.
- ACK: useful as it indicates if the acknowledgment number field is valid or not
- Checksum: used for error detection.

Let us describe briefly the importance of these critical parameters. Here, the Source port number, the Destination port number, the Checksum fields have almost the same purpose as in UDP. One thing that we can add is to stress the importance of the Checksum field which performance impacts the reliability of TCP. Therefore is very critical here. Talking about the Sequence number field, we can say that this field is critical in the sense that if helps to site the receive data in the flow of information, any undetected error in this field means that the flow of information may be altered or the data will be discarded if error leads to a smaller sequence number than the one expected. Finally the bits ACK, RST, SYN, FIN are critical since they concern the management of the communication process. Let us look at non-sensitive control information.

Non-sensitive control information:

In this sub-section, we are going to talk about TCP header fields that we consider to be non-sensitive with regard to reliability and we will explain the reason of our choice. One field that, we consider as non-sensitive is the window one. The window's data is important for flow control but we choose to consider it as non-critical information for the good functioning of the reliable protocol. Indeed an error in this field will not be harmful for the reliable process, it will influence mostly the flow control procedure. Moreover, this error can be alleviated by subsequent transmissions which can provide error-free information. Another non-critical field for reliability is the acknowledgment number field. Again, the acknowledgment number field determines the way the information will be sent back by the recipient. However, the acknowledgment number is not consider as a critical parameter since TCP used cumulative acknowledgment, an error in this field will not be harmful for the functioning of the reliability process. Indeed, subsequent packets with correct acknowledgment number field will restore the information sent or needed. In addition to these non-sensitive information, we consider in this paper that the bits PSH, URG are not used and that throughout the transmission the size of the header field remains constant i.e. 20 bytes. As for UDP, we do not plan to protect non-sensitive control information with a coding. However one can derive from this work an algorithm which may include some of

these bits or fields. Table 5 summarizes the classification and assumption we are using for the rest of the paper. Having identified the critical fields in the header part of TCP the next step is to improve their reliability.

Table 5. Summary of the classification and assumption for TCP.

Sensitive control data	Non-sensitive control data
• Source port number • Destination port number • Sequence number • Checksum • A • R • S • F	• acknowledgment number • Receiving window • Data-offset (not-used) • Not-used (not-used) • U (not-used) • P (not-used) • Urgent pointer (not-used) • Options (not-used)
Size of the TCP header 20 bytes	

3.2. Algorithm for Coding Some Control Data for TCP

This section deals with the algorithm we are suggesting. The idea is to encode the sensitive data of TCP header. We also make use of habits of users when they go to the Internet as mentioned earlier in section 2. Based on this observation we decide to rescale the number of bits use for source port number and destination port number in the same way as for UDP (see section 2). The bits of the PSH and URG are used for coding. Remember that the data that we are encoding are: source port number, destination port number, sequence number, RST, SYN, FIN, ACK and checksum. To distinguish between the four states we suggest in this case to use 5 bits sequences with a minimum Hamming distance of 3 as presented in Table 6.

Table 6. Example of assignment between size-state and bits.

5 bits used	purpose
00010	The size of the data field is MSS and the state is 1
00101	The size of the data field is MSS and the state is 2
11000	The size of the data field is MSS and the state is 3
11111	The size of the data field is MSS and the state is 4

From this table, we have derived Table 7, which gives the details of what we obtain from each state while providing the coding rate of each state. Let us make the remark that a possibility of including the checksum in the decoding process may exist since a first decoding provides the checksum result. Then, any error detected by the checksum can be used again in a second decoding process in order to improve the reliability of bits (principle similar to turbo code). This procedure could have the effect similar to having smaller code rate.

Table 7. TCP: states and coding rate.

States	Length of control data to encode	Length of header field Except acknowledgment number, receive window, option	Coding rate
1	4 + 52	112	56/107
2	8 + 52	112	60/107
3	16 + 52	112	68/107
4	32 + 52	112	84/107

Let us sum up the method which captures the proposed algorithm.

METHODOLOGY:

1) Discriminate between sensitive and non-sensitive control data.

2) Select amongst the non-sensitive control data those which are usually not-used or which functionalities can be handled by another mechanism (example Length field replaced by *MSS*).

3) Use the fields selected in 2. to code sensitive fields.

Having presented the algorithm we are using to encode some control data in TCP and UDP header let us see which potential gain we can have with such code rate.

4. Results and Discussion

This section is indented for illustrating the gain that we can obtain in terms of BER (Bit error rate) while using our algorithm. It worth noticing that this gain concerns only the sensitive control data. As subsection 2.2 and subsection 3.2 show the coding rate can be less or close 1/2. Thus we choose to exemplify the gain by using a convolutional code [10] with the following characteristics:

- a (2, 1) convolutional code.
- coding rate: 1/2.
- generator sequences g1 = [101] and g2 = [111].

The goal here is to have an estimate of the potential gain we can have by coding the sensitive control data bit error rate. Moreover, we model the Transport layer channel as Binary Symmetric Channel [11] with a probability of error p. It is the probability that a given bit sent 0 or 1 is reversed by the channel. Figure 3 illustrates this logical view of the transmission of data between transport layer entities. The logical channel physically represents all the transfer and processing that occur between the moment a TCP segment is created and the point where it is received on the recipient side. This includes host processing (on both end systems involved in the communication) by network layer, data link layer, physical layer and the different processing by the packet switches on the path followed by the packet containing the segment. As far as burst errors are concerned techniques such as

bit-interleaving can be used to convert a channel prone to burst errors into a channel which follows a binary symmetric law. Moreover we will also keep in mind that there is a link between packet error and bit errors.

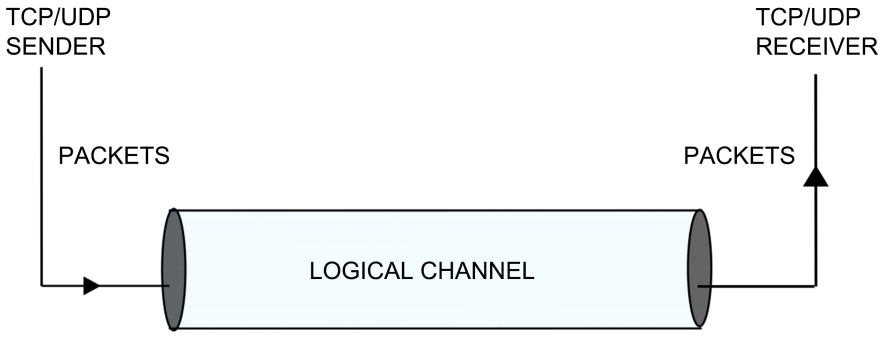

Figure 3. Transport layer channel model.

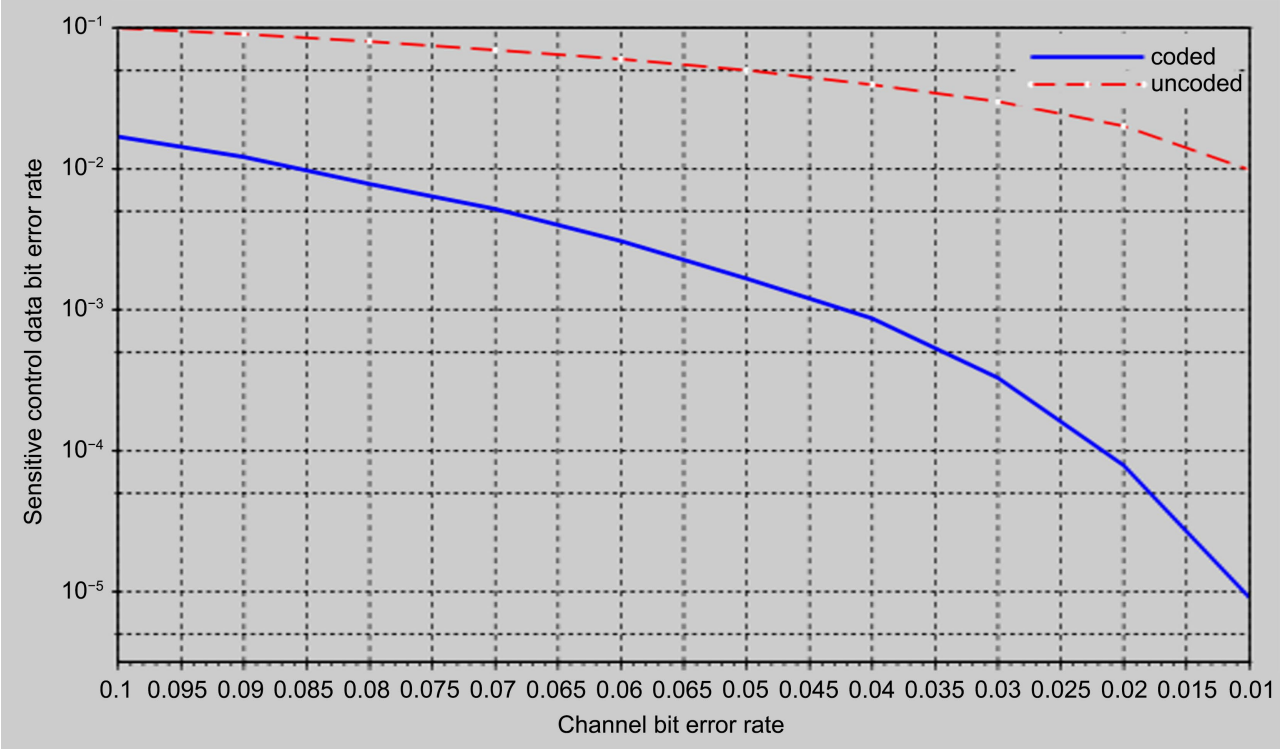

Figure 4. Sensitive control information bit rate.

Then, Figure 4 displays two curves corresponding to the result of our simulation. The dash line one corresponds to the sensitive control data bit error rate as a function of the channel bit error rate without coding (uncoded). The solid line is the result of the sensitive control data bit error rate as a function of the channel bit error rate with coding (coded). As we can notice the coding of the sensitive control data yields a factor of around 10^{-2} in terms of bit error rate between coded and uncoded data. Another remark one can draw up from this plot is that, the performance of the uncoded data follows the channel bit error values which is

consistent. The improvement in the sensitive control data means for example that for packets arriving at the recipient side with errors in their destination port number, the coding will be able to correct in many cases all these errors therefore the packets will be delivered to the right process instead of being discarded. This will avoid some retransmissions mainly the case of TCP or loss of packets in the case of UDP. Let us recall that avoiding retransmission means saving power as sending bits requires energy. Moreover, it will improve the welfare of networks by reducing the load processed by the core of networks. This last point suggests a positive impact in terms of alleviating congestion in the network.

5. Conclusions

In this paper, we developed an approach in order to encode some fields of the header part of TCP and UDP segments. To achieve that: Firstly, we classified the header fields in two classes according to fact that they are more or less relevant:

- For UDP, the priority was given to the identification of the sender and recipient (port numbers) and the performance of the checksum mechanism.
- For TCP, the priority was given to the identification of the sender and recipient, the performance of the checksum mechanism and all relevant information for connection set up and tear down. Secondly, we have suggested using some fields for the encoding process after that, we have taken into account the fact that for many users the size of the field assigned for port numbers can be big. Thus we proposed re-scaling these fields and using the remaining bits for encoding. Finally, we used convolutional code to illustrate the performance of this method. Here, the approach was to have both sides of the communication in the same state which can evolve throughout a given communication. Some studies could be done in the case we decide that both sides of the communication can be in two different states. We can also investigate the possibility of including the checksum either partially or totally in the design of some coding scheme for better performances.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Postel, J. (1981) ISI. Request for Comments 793, Transmission Control Protocol, Darpa Internet Program Protocol Specification, September.

[2] Duke, M., Braden, R., Eddy, W., Blanton, E. and Zimmermann, A. (2015) IETF. Request for Comments 7414—A Roadmap for Transmission Control Protocol (TCP) Specification Documents, February.

[3] Kurose, J.F. and Ross, K.W. (2000) Computer Networking—A Top-Down Approach Featuring the Internet. Addison-Wesley.

[4] Stevens, W.R. (1994) TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley.

[5] Comer, D.E. (1991) Internetworking with TCP/IP: Principles, Protocols, and Architecture, Volume I. 2nd Edition, Prentice-Hall.

[6] Postel, J. (1980) ISI. Request for Comments 768—User Datagram Protocol, 28 August.

[7] Eggert, L., Fairhurst, G. and Shepherd, G. (2017) IETF. Request for Comments 8085—UDP Usage Guidelines, March.

[8] Blahut, R.E. (1983) Theory and Practise of Error Control Codes. Addison-Wesley.

[9] Lin, S. and Costello, D.J. (1983) Error Control Coding: Fundamentals and Applications. Prentice-Hall.

[10] Proakis, J.G. and Salehi, M. (2001) Communication Systems Engineering. 2th Edition, Eastern Economy Edition.

[11] Cover, T.M. and Thomas, J.A. (1991) Elements of Information Theory. Wiley Series in Telecommunications.