**TOPIC :CHECKSUM VERIFICATION IN DATALINK LAYER USING RAW SOCKETS**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA**

**YEAR-2017-18**

**COURSE TITLE**

Computer Communication & Networking (IT251)

**SUBMITTED TO :**

Mrs Geetha V

Course Instructor

**SUBMITTED BY :**

Supriya Sahoo (15IT145)

Akriti Kumari (15IT107)

Ankita Bhalavi (15IT108)

Vandana Baidya (15IT250)

**CERTIFICATE**

This is to certify that the project entitled **“**Checksum verification in datalink layer using raw sockets**”** has been presented by Supriya Sahoo, Akriti Kumari, Ankita Bhalavi, Vandana Baidya, students of second year, B.Tech (IT), Department of Information Technology, National Institute of Technology Karnataka, Surathkal, on April \_2017, during the even semester of the academic year 2016- 2017, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology at NITK, Surathkal.

Place: NITK, Surathkal

Date: 01/05 2017 (Signature of the Examiner)

**ACKNOWLEDGEMENT**

This project would not have been possible without the kind support and help of many

individuals and organizations. We would like to extend our sincere thanks to all of them. We are highly indebted to the Department of Information Technology, NITK for their guidance and constant supervision as well as for providing necessary information and resources for the project and also for their support in completing the project.

We would like to express our gratitude towards our teacher Mrs Geetha V for her kind co-operation and encouragement which helped us in completion of this project. Our thanks and appreciations also go to our group members in developing the project and people who have willingly helped us out with their

abilities.

**ABSTRACT**

This project aims at verifying the checksum calculation in datalink layer using raw sockets. A *checksum* is a simple type of redundancy check that is used to detect errors in data.Errors frequently occur in data when it is written to a disk, transmitted across a network or otherwise manipulated. The errors are typically very small, for example, a single incorrect bit, but even such small errors can greatly affect the quality of data, and even make it useless.

In its simplest form, a checksum is created by calculating the binary values in a packet or other block of data using some algorithm and storing the results with the data. When the data is retrieved from memory or received at the other end of a network, a new checksum is calculated and compared with the existing checksum. A non-match indicates an error; a match does not necessarily mean the absence of errors, but only that the simple algorithm was not able to detect any.

**TABLE OF CONTENTS**

1.Introduction……………………………………………………………………………6

1.1 Method…………………………………………………………………………..7

2. Implementation Details……………………………………………………………….8

3. Result………………………………………………………………………………….

4. Conclusion ……………………………………………………………………………

5. References…………………………………………………………………………….

**INTRODUCTION**

A checksum is an error-detection method in a the transmitter computes a numerical value according to the number of set or unset bits in a message and sends it along with each message frame. At the receiver end, the same checksum function (formula) is applied to the message frame to retrieve the numerical value. If the received checksum value matches the sent value, the transmission is considered to be successful and error-free.  
  
A checksum may also be known as a hash sum.

A mismatched checksum shows that the entire message has not been transmitted. TCP/IP and User Datagram Protocol (UDP) provide a checksum count as one of their services.  
  
The procedure of generating checksums from messages is called a checksum function and is performed using a checksum algorithm. Efficient checksum algorithms produce different results with large probabilities if messages are corrupted. Parity bits and check digits are special checksum cases suitable for tiny blocks of data. Certain error-correcting codes based on checksums are even capable of recovering the original data.   
  
The most commonly-used checksum tools include:

* "cksum" - Unix commands generating 32-bit cyclic redundancy check (CRC) and byte count for an input file
* "md5sum" - Unix command generating Message-Digest Algorithm 5 (MD5) sum
* "jdigest" - Java GUI tool generating MD5 and Secure Hash Algorithm (SHA) sums
* "Jacksum" - Java application programming interface that incorporates numerous checksum implementations and permits any number of extensions
* "jcksum" - Java libraries used to calculate checksum using different algorithms

Several network protocols use checksums to ensure data integrity. Applying checksums as described here is also known as redundancy checking.

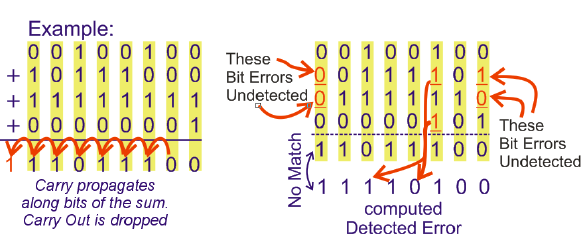
Checksums are used to ensure the integrity of data portions for data transmission or storage. A checksum is basically a calculated summary of such a data portion.

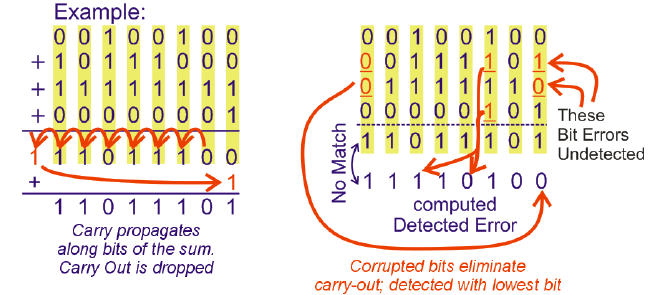
**1.1 METHOD**

Network data transmissions often produce errors, such as toggled, missing or duplicated bits. As a result, the data received might not be identical to the data transmitted, which is obviously a bad thing.

Because of these transmission errors, network protocols very often use checksums to detect such errors. The transmitter will calculate a checksum of the data and transmits the data together with the checksum. The receiver will calculate the checksum of the received data with the same algorithm as the transmitter. If the received and calculated checksums don’t match a transmission error has occurred.Some checksum algorithms are able to recover (simple) errors by calculating where the expected error must be and repairing it.

If there are errors that cannot be recovered, the receiving side throws away the packet. Depending on the network protocol, this data loss is simply ignored or the sending side needs to detect this loss somehow and retransmits the required packet(s).Using a checksum drastically reduces the number of undetected transmission errors. However, the usual checksum algorithms cannot guarantee an error detection of 100%, so a very small number of transmission errors may remain undetected.There are several different kinds of checksum algorithms; an example of an often used checksum algorithm is CRC32. The checksum algorithm actually chosen for a specific network protocol will depend on the expected error rate of the network medium, the importance of error detection, the processor load to perform the calculation, the performance needed and many other things.





**IMPLEMENTATION DETAILS**

**RESULT**

**CONCLUSION**

The algorithm for the datalink layer checksums is identical. The data is processed a word (16 bits, two bytes) at a time. (An odd number of bytes just has a trailing zero byte added to make the total number even.) The words are added using ones-complement arithmetic, with the location holding the checksum set to be zeros. Once this long chain of additions is complete, the result is negated in ones-complement by taking the binary not, and the result is stored in the right spot. If this operation is repeated then the result of the checksum should be the binary all-ones.

The algorithm has many interesting properties. The first is that since it is based on addition, it is both commutative and associative. This means that we can use some other calculation that effectively adds in some other order. By re-arranging the additions we might be able to increase performance.

The second property is more subtle: The result is endian independent. A ones-complement addition can be performed on twos-complement machines (most of them) by doing a normal twos-complement addition, followed by adding the potential carry back in the least-significant position. This means that we effectively have a carry from the first byte into the second, and from the second byte into the first. This symmetry means that no matter what endian order we internally place on those two bytes, the calculation will give the same result.

A final interesting property (not really used in this article) is that since the checksum is based on addition, we can alter a packet and update the checksum cheaply. We just need to do a ones-complement subtraction of the differences in the words we change. For example, this means that when a router updates the time-to-live (TTL) of a packet, it doesn't have to redo the whole checksum calculation.

Evaluating the checksum can be a bottle-neck in network heavy environments. This is why many network interfaces offer the ability for the hardware to do the resulting checksum calculations.

**REFERENCES**

* https://en.wikipedia.org/wiki/Data\_link\_layer
* http://ieeexplore.ieee.org
* http://www.online-tech-tips.com
* https://en.wikipedia.org/wiki/Checksum
* https://locklessinc.com/articles/tcp\_checksum/