Journal of Science Engineering Technology and Management Science
Volume 02, Issue 08, August 2025 *ISSN: 3049-0952*
www.jsetms.com *DOI:10.63590/jsetms.2025.v02.i08.420-427*

DATA COMPRESSION FOR BACKBONE NETWORK SECURITY

[1] M. SUJITH,[2] T. BHARATH REDDY,[3] P. ASHOK,[4] T. SIDHARTHA REDDY,[5] Mr. R. BHARATH
[1234] Students, [5] Assistant Professor
Department Of Computer Science and Design
Teegala Krishna Reddy Engineering College, Meerpet, Balapur, Hyderabad-500097

To Cite this Article
M. Sujith, T. Bharath Reddy, P. Ashok, T. Sidhartha Reddy, Mr. R. Bharath, "Data Compression For Backbone Network Security", Journal of Science Engineering Technology and Management Science, Vol. 02, Issue 08, August 2025,pp: 420-427, DOI: http://doi.org/10.63590/jsetms.2025.v02.i08.pp420-427
Submitted: 12-07-2025 *Accepted: 18-08-2025* *Published: 25-08-2025*

ABSTRACT
This paper presents a review kind of data compression ways. Data compression is extensively used by the community because through a compression we can save storehouse. Data compression can also speed up a transmission of data from one person to another. In performing a compression requires a system of data compression that can be used, the system can also be used to compress a data. Data that can be compressed not only textbook data but can be images and videotape. Data compression fashion is divided into 2 videlicet lossy compression and lossless compression. But which is frequently used to perform a compression that's lossless compression. A kind of lossless condensing similar as Huffman, Shannon Fano, Tunstall, Lempel Ziv welch and run- length encoding. Each system has the capability to perform a different compression. This paper explains how a system works in doing a compression and explains which system is well used in doing a data compression in the form of textbook. The affair generated in doing a can be known through the compression train size that becomes lower than the original train.

I. INTRODUCTION

With the rapid-fire development of technology with the support of software and tackle that decreasingly grease wide information snappily through the internet around the world. Information attained can be transferred fluently via the internet as the medium of communication for information technology experts. still, not all information can be transferred fluently. There's a large size that can hamper data transmission snappily and save on being storehouse in the computer. To overcome the problem of information or data to be transmitted or transmitted can be done snappily than needed a compression that can save storehouse and transmission of data to be done. Compression is the process of converting a data set into a law to save the need for storehouse and transmission of data making it easier to transmit a data. With the compression of a can save in terms of time and storehouse that live in memory(storehouse). numerous compression algorithm ways can be performed and function duly similar as the Huffman, Lempel Ziv Welch, Run Length Encoding, Tunstall, And Shannon Fano styles. The data process of data compression is shown in figure 1.

Figure 1: The data process of data compression

In figure 1, explain the process of data compression in general. how the data when not compressed also uncompressed data will be continued and reused by compression system that's lossless compression also the data has been compressed will produce a size lower than the size of the train before it's compressed. Compression is the reduction of a train size from a large size to a lower train size. A compression will be done to grease the transmission of a train with a large size and contains numerous characters. The workings of a compression are by looking for patterns of reiteration in the data and replace it with a certain sign. The type of compression has two styles, lossless compression and lossy compression. Lossless compression is the process of converting the original data with compressed data becomes further terse without reducing the loss of information. Lossy compression is the prose converts the original data into the compression data there are different values, but the value of this difference is considered doesn't reduce the essential information from the original data. Then's an explanation of the connection that exists in a data compression Compression for audio

- Compression for text
- Compression for video
- Compression for image
- Compression for audio Audio

1.1 MOTIVATION

Data compression plays a pivotal part in enhancing backbone network security by perfecting both effectiveness and protection. In large- scale networks, similar as those used by internet service providers and data centers, enormous volumes of data are transmitted every second. Without compression, this can lead to traffic, increased quiescence, and lesser exposure to cyber pitfalls. Compressing data reduces the size of the transmitted information, which helps in minimizing bandwidth consumption, lowering transmission costs, and perfecting the overall speed and performance of the network. From a security viewpoint, data that's compressed is frequently less readable and more delicate to manipulate, adding an fresh hedge for unauthorized druggies. likewise, compression can be integrated with encryption and other security protocols to form a multi-layered defense system, making it harder for bushwhackers to block or alter the data. Reduced data volume also decreases the duration for which sensitive information is in conveyance, thereby limiting the window of occasion for interception. In substance, data compression not only optimizes network performance but also strengthens data confidentiality, integrity, and vacuity — crucial pillars of cybersecurity in backbone architectures.

1.2 PROBLEM STATEMENT

In backbone networks, the exponential growth of data business due to the adding demand for internet services, pall computing, and real- time operations poses significant challenges to network effectiveness and security. Traditional data transmission styles frequently affect in high bandwidth operation, network traffic, and increased quiescence, which can compromise the speed and trustability of data delivery. also, large volumes of uncompressed data in conveyance present a broader attack face for cyber pitfalls similar as data interception, tampering, and denial- of- service(DoS)attacks. There's a critical need to apply effective data compression ways that not only reduce transmission cargo but also enhance the security of data covering these high- capacity networks. This exploration aims to explore how data compression can be strategically abused to ameliorate both performance and security in backbone network architectures.

1.3 SCOPE AND OBJECTIVE

The compass of this study focuses on probing the part of data compression ways in enhancing the

performance and security of backbone networks. It covers the analysis of different compression algorithms, their integration with encryption styles, and their impact on network outturn, quiescence, and vulnerability to cyberattacks. The work is limited to large- scale backbone networks generally used by internet service providers, pall architectures, and enterprise data centers. It aims to estimate the trade- offs between compression effectiveness, computational outflow, and security strength, furnishing recommendations for optimized data transmission without compromising network security norms.

Objectives:

- To dissect the challenges faced by backbone networks in handling massive data volumes securely and efficiently.
- To study colorful data compression ways and estimate their effectiveness in reducing network cargo.
- To examine the relationship between data compression and network security, particularly how compression affects data confidentiality, integrity, and vacuity.
- To design or propose a compression-grounded frame that integrates security features to cover data during transmission across backbone networks.
- To assess the performance of the proposed result in terms of bandwidth application, transmission speed, and resistance to cyber pitfalls.
- To give guidelines and stylish practices for planting compression strategies that enhance both performance and security in backbone network architectures.

II. LITERATURE SURVEY

Run Length Encoding:

RLE (Run Length Encoding) algorithm is one algorithm that can be used to compress data so that the size of the data produced is lower than the actual size. The example discussed this time is the cost and return of data from a sentence.

RLE (Run Length Encoding) is the easiest form of lossless data compression technique where a series of data with the same value in sequence will be saved into a data. This algorithm is very useful in data that has a lot of data with the same values in sequence like icon files, line drawings, and animations. This algorithm is not suitable for normal data because it will increase.

Lempel Ziv Welch:

In general the LZW algorithm is a lossless compression algorithm and uses a dictionary. LZW compression will form a dictionary during the compression process takes place.. LZW algorithm can be implemented in many compression applications. In this experiment [5] have provided a comparison between the conventional LZW coding and proposed MLZW coding [5]. Compression result in term of dictionary. Output from LZW algorithm is amount of bit or codeword compression result must be small than file before compression. Algorithm is adapted for Unicode standard, it may be used very easily for any Bangla compression text [5].

Concept from this algorithm is to find the new dictionary from a new character. LZW method use variable word width dictionary to balance the compression and decompression file.

Tunstall:

The Tunstall algorithm was the subject of Brian Parker Tunstall's thesis in 1967 while at the Georgia Institute of Technology. The subject of this thesis is "Synthesis of noiseless compression codes." The Tunstall Algorithm is a code that maps the source symbol to a fixed number of bits as well as sorts the stochastic source with a variable length codeword.

Huffman:

The Huffman method applied in this paper is a static type, which is done twice (two-pass) reading of a data / file to be compressed. To calculate the appearance of characters in the formation of a Huffman tree and encode with the Huffman code symbol.

This method encodes symbols or characters with the help of a binary tree by combining the two smallest frequencies to form a code tree. Huffman codes are based on the number of character frequencies that often appear.

The larger the Huffman code frequency the less number of bits produced. Conversely, the fewer character appearances the more number of bits produced during a compression. This Huffman compression algorithm includes methods lossless compression. lossless compression is a compression technique that does not change the original data information to a smaller size.

The rationale of Huffman's algorithm is that each ASCII character is usually represented by 8bits. So for example a file contains a row of characters "AABCD" then the number of bits in the file 5 x 8 is 40 bits or 5 bytes.

If each character is given a code eg A = 0, B = 10, C = 111, D = 110 then we only need a file with the size of 10bits (0010111110). That note that the codes must be unique or in other words a code can not be formed from other codes.

EXISTING SYSTEM

In the current backbone network infrastructures, data transmission primarily focuses on achieving high speed and reliability, with security measures applied separately through encryption protocols such as SSL/TLS, IPSec, and VPNs. While these security mechanisms provide strong protection against interception and unauthorized access, they often introduce additional overhead, leading to increased latency and bandwidth consumption. Traditional compression techniques like GZIP, LZ77, and Huffman coding are used in certain scenarios to reduce data size, but they are typically implemented at application layers (e.g., web servers, storage systems) rather than being deeply integrated into the network's core operations. Furthermore, compression and security are generally treated as independent processes, rather than as a combined strategy.

Limitations of the Existing System:

- High Overhead: Existing encryption methods, while effective for securing data, often introduce additional processing delays and increase bandwidth usage, affecting overall network performance.
- Separate Handling of Compression and Security: Compression and security are typically handled independently, missing the opportunity to create an integrated system that optimizes both simultaneously.
- Limited Scalability: Traditional compression techniques used in the current system are not optimized for the massive and continuously growing data loads seen in backbone networks, leading to scalability issues.

PROPOSED SYSTEM.

A proposed system typically involves improvements or optimizations over the existing one. Based on the review of the document, some suggestions might include:

1. Enhanced Lossless Compression Algorithms: Incorporating improved versions of traditional algorithms, such as Modified LZW (MLZW), which adapts for specific languages like Bangla, as suggested in the document. These improvements can lead to higher compression rates and efficiency.
2. Adaptive Compression Systems: Developing systems that dynamically choose between lossy and lossless methods based on data type, real-time requirements, and user needs. For instance, combining Huffman coding with other methods to improve the overall compression rate.
3. Parallel Processing for Compression: To speed up the compression process, especially for large files like video or high-definition images, leveraging parallel computing or machine learning techniques could improve the efficiency and speed of compression without sacrificing quality.
4. Compression Techniques Optimized for Modern Use Cases: With the growing use of IoT devices and cloud storage, compression systems need to be lightweight and resource-efficient,

tailored to limited bandwidth and processing power constraints.

Advantages:

1. Improved LZW (MLZW): Achieves higher compression rates by adapting to specific language characteristics.
2. Adaptive Compression: Dynamically selects the optimal compression method for each data type, maximizing efficiency.

Parallel Processing: Speeds up the compression process, making it more suitable for large datasets like high-definition video.

III. MODULE DESCRIPTION

1. User Registration Module Registration Form: A user-friendly form that collects essential details like: Username Password (with encryption) Email (with verification) Optional profile information (e.g., full name, phone number). Validation: Ensure mandatory fields are filled. Password strength check (minimum length, complexity). Email uniqueness check (to prevent duplicates). Account Creation: Once validated, create an account, store credentials securely (e.g., password hashing). Send a confirmation email to the user's registered email.

2. User Login Module Login Form: Collect username/email and password. Offer 'Remember Me' option for maintaining user sessions. Authentication: Check username/password combination against the database. Implement login attempt limits to prevent brute force attacks. Session Management: Create a session for the user upon successful login. Provide automatic logout functionality after a period of inactivity.

3. Upload Document Module File Upload Interface: Users can select and upload documents in various formats (e.g., PDF, DOCX, TXT). Implement file size restrictions and format validations. Document Storage: Store documents in a secure file system or cloud storage. Ensure unique naming conventions to avoid overwriting existing files.

Meta Information: Collect and store metadata about the uploaded document (e.g., filename, size, upload date).

4. Zip Document Module Zip Functionality: Allow users to select multiple uploaded files for zipping. Automatically compress the selected files into a ZIP archive. Customization: Allow users to name the ZIP file. File Integrity: Ensure the zip file is created without any corruption. Show the size of the zipped file to the user.

5. Upload Zip File to Server Module File Selection and Upload: Allow users to upload ZIP files directly to the server. Validate that the file is a valid ZIP archive.

Server Storage: Store the uploaded ZIP file securely. Display upload progress to the user.

6. View Documents Module Document List View: Display all documents uploaded by the user in a list or grid format. Provide relevant metadata (filename, size, upload date). Filtering and Sorting: Allow sorting by file type, upload date, size, etc. Provide filtering options to show specific document types. Document Preview: Enable preview of certain document types (e.g., PDF, images) without downloading.

7. Extract Zip File Module Unzipping Functionality: Users can select a ZIP file and extract its contents on the server. Display the list of extracted files upon completion. Error Handling: Handle issues like corrupted ZIP files with appropriate error messages. Destination Management: Allow users to choose where the extracted files should be stored. Overwrite or duplicate handling for extracted files with the same name.

8. Download Extracted File Module File Selection: Provide an interface to select one or more files from the extracted contents. Download Functionality: Allow users to download individual files or multiple files as a ZIP. Implement download progress indicators. Security: Ensure only authorized users can download files. Optionally encrypt files before download for additional security.

9. Delete Document Module File Deletion: Allow users to delete uploaded or extracted

documents. Confirmation: Prompt for confirmation before deleting files to avoid accidental deletions. Permanent or Temporary Deletion: Option to send files to a 'Trash' where they can be recovered later or permanently delete them from the system.

10. Search Document Module Search Interface: Provide a search bar where users can search for documents by name, date, type, or other metadata. Advanced Search: Enable advanced search options like searching within document contents (e.g., text search for PDF or DOCX) Search Results: Display matching documents with relevant metadata and preview options.

IV. SYSTEM DESIGN

SYSTEM ARCHITECTURE

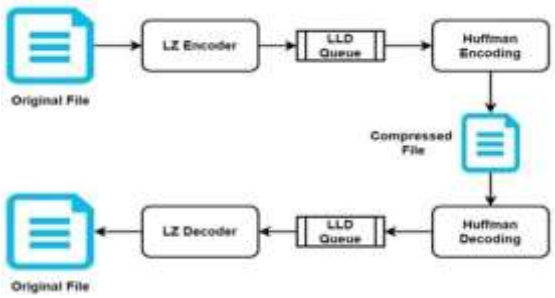

Fig. System Architecture

V. OUTPUT SCREENS

Fig: Selecting run on server

Fig: Using Apache tomcat server

Fig: Home page

Fig: Registration

Fig: View Document

Fig: Sql yog

VI. CONCLUSION

In conclusion, data compression plays a critical role in enhancing backbone network security by improving the efficiency, speed, and reliability of data transmission. By reducing the size of data, compression not only optimizes bandwidth usage but also minimizes the risk of congestion and delays, which are crucial for maintaining secure and high-performance networks. Additionally, compression can complement encryption and other security mechanisms by making it harder for attackers to intercept and manipulate large volumes of sensitive information. However, it is important to carefully select and implement compression techniques that do not compromise data integrity or open new vulnerabilities. Overall, integrating efficient data compression strategies strengthens the security posture of backbone networks, ensuring faster, safer, and more resilient communication across critical infrastructures.

Using the compression technique can reduce the number of file sizes. data that has a large size into a smaller size that can save storage in a computer. data compression can be implemented on a text, photo, and video data. various compression algorithm techniques have advantages and disadvantages in doing a compression.

REFERENCE

[1] Xiaoyu Ruan, Rajendra Katti, "Using Improved Shannon-Fano-Elias Codes Data Encryption", Proceedings of ISIT Conference, North Dakota State University Fargo, July 9-14, 2006.

[2] Mr.Mahesh Vaidya, Mr.Ekjot Singh Walia, , and Mr.Aditya Gupta, "Data Compression Using Shannon Fano Algorithm Implemented By VHDL", IEEE International Conference on Advances in Engineering & Technology Research, August 01-02,2014.

3] Lung-Jen Lee, Wang-Dauh Tseng, Rung-Bin Lin, and Cheng-Ho Chang, " Pattern Run-Length for Test Data compression", IEEE Transaction on Computer-Aded Design of Integrated Circuits And System, Vol.31, No.4,April,2012.

4] Mohammad Arif, R.S.Anand, "Run Length Encoding for Speech Data Comprassion", IEEE International Conference on Computional Intelligence and Computing Research, 2012.

5] Linkon Barua, Pranab Kumar Dhar, Lamia Alam, and Isao Echizen, " Bangla Text Compression Based on Modified Lempel-Ziv-Welch Algorithm", International Conference on Electrical, Computer and Communication Engineering (ECCE), Bangladesh, February 16-8,2017.

[6] International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8956-8963 © Research India Publications. http://www.ripublication.com G.R.Gnana King, C.Seldev Christoper, And N.Albert Singh, " Coumpound Image Compression Using Parallel Lempel Ziv-Welch Algorithm", IET Chennati Fourth International Conference on Sustainable Energy and Intelligent System,Chennati,December 12-14, 2013.

[7] Haoqi Ren, " A data Comprssion Technique based on Resersed Leading Bits Coding and Huffman Coding", International Conference on Communication and Networking, China, 2015.

[8] Haoqi Ren, " A data Comprssion Technique based on Resersed Leading Bits Coding and Huffman Coding", International Conference on Communication and Networking, China, 2015. Djuned Fernando Djusdek, Hudan Studiawan, and Tohari Ahmad, " Adaptive Image Compression Using Adaptive Huffman and LZW", International Conference on Information, Communication Technology and System, 2016. Tsutomu Kawabata, " Enumerative Implementation of Lempel-Ziv-77 Algorithm", ISIT, Toronto, Canada, July 6-11, 2008.

[9] Adrian Traian Murgan, Radu Radescu, " A Comprison of Algorthm for Lossless Data Compression Using the Lempel-Ziv-Welch Type Methods", Bucharest.

[10] Victor Amrizal, " Implementasi Algoritma Kompresi Data Huffamn Untuk Memperkecil Ukuran File MP3 Player", 2-14,2010.

[11] Cut Try Utari, " Implementasi Algoritma Run Length Encoding Untuk Perancangan Aplikasi Kompresi dan Dekompresi File Citra", Jurnal TIMES, Vol.V No.2, 24 31, 2016.

[12] M.VidyaSagar, J.S, Rose Victor, "Modified Run Length Encoding Scheme for High Data Compression Rate", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Vijayawada, December 2013. IEEE International Nanotechnology, Conference Information Humanoid, Technology Communication and Control, Environment and Management (HNICEM). Philippines, 9-12 Decmber 2015.

[13] K. Ashok Babu and V. Satish Kumar, "Implementation of Data Compression Using Huffman Coding", International Conference on Methods and Models in Computer Science, India, 2010.

[14] Harry Fernando, "Kompresi data dengan algoritma Huffman dan algoritma lainnya", ITB, Bandung.