Squeeze Pack and Transfer Algorithm: A new over the top Compression Application for Seamless data Transfer over Wireless Network

Amandeep Bagga, Shiv Preet

Abstract: ***Mobile data transfer is the backbone of Information Technology industry. It is now becoming the daily bread and butter earning means for most of the third world and developed countries. This paper proposed a text based compression algorithm which can be used with existing algorithms as an OTT (over the top) service and improves their functionality. It can take advantage of increasing processing power of new generation CPUs like Snapdragon and Mediatechas well as bandwidth efficiency of modern mobile infrastructure. This algorithm is based on .Net Assemblies of Microsoft Visual Studio 2017 (Community Edition) and VB.Net (Visual Basic.Net) language. This algorithm can help in improving data transfer speed over wireless and mobile networks. It can also secure data transfer for vulnerable attacks (Man in the Middle Attack, DDOS Attack etc.). It will also help in reducing network congestion as it can pack more data per packet as compared to traditional algorithms.***

Keywords: ***OTT, C#, .Net, Polymorphism, IPTV, Compression, Decompression, Key Pool, Encryption, Decryption, Lossless Compression, Lossy Compression, Winzip, Winrar, DDOS.***

Introduction

Compression technologies have very important place in telecom sector. They help in increasing the volume of data transfer without any change in current network infrastructure [1].Lossless compression technologies are needed where data accuracy is of prime importance. It is observed that now a days even streaming sites are trying to use lossless compression, especially IPTV or Internet protocol televisions requires such techniques. IPTV industry is booming with new content player entering into it almost every day [2].

HBO go, Sky Go, MOBDRO and other such services relies heavily on good Internet connection. Streaming services like Netflix, Amazon Prime, Hot Star etc. are now replacing the conventional television viewing system [3]. A 4K streaming of Netflix demands at least 25 mega bit per second speed on an Internet connection with very low latency. In mobile space, it is very difficult to provide such higher speed in a consistent manner [4].

Revised Manuscript Received on May 23, 2019.
Shiv Preet, Ph.D. Scholar, Lovely Professional University
Dr. Amandeep, Associate Professor and Deputy Dean, Lovely Professional University

Internet speed in wireless arena depends on many factors like number of users currently connected, availability of signals, Base Trans-receiver System available at a particular location etc. Compression technologies can work wonder in providing seamless experience to users in wireless network infrastructure. In this paper one such technique has been described [5]. It has been measured quantitatively as well as qualitatively that how this technique reduces size of file without reducing its quality before sending it to a remote location [6]. It has also been compared that how various popular algorithms like Zip, Rar and 7-Zip fares without this technique and what improvement will be obtained if this technique is used as an over the top service for these algorithms [7].

Basic Algorithm

- Select file **F** which you want to send over the network .
- Find similar patterns of binary Keys (**BK**) in the file.

  BK={k_1,k_2,k_3,...k_n}.

- Create a keyfile (**KS**) and send it to receiver over the network.
  KS={ks_1,ks_2, ks_3, ...ks_n}.
- Split selected file **F** into small subparts (**FS**).

  FS={fs_1,fs_2, fs_3, ...fs_n}.

- Compress each (**Fs**) using binary key file (**Ks**) into (**FSK**).

  FSK={fsk_1,fsk_2, fsk_3, ...fsk_n}.

- Compress each (**Fsk_t**) into (**$Fskk_t$**) using any generic lossless compression or any third party application where **Fsk_t** is t_{th} symbol in the set **FSK** and **$Fskk_t$** is t_{th} symbol in the set **FSKK**.

  FSKk={$fskk_1$,$fskk_2$, $fskk_3$, ...$fskk_n$}.

- Send (**$Fskk_t$**) to receiver over the network where **$Fskk_t$** ist$_{th}$ symbol in the set **FSKK**.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

- Decompress (**$Fskk_t$**) to (**Fsk_t**) using generic lossless compression or any third party application (application should be same which were used while compressing the file.
- Decompress (**Fsk_t**) to (**Fs_t**) using binary key file (**KS**).
- Rejoin each (**FS**) to create source file **F** at recipient node.

Implementation

Its detailed implementation has been divided into two parts for the purpose of this research paper. In phase 1 processes at sender side has been explained and in phase 2 processes at receiver side has been explained.

Processes at sender side (Phase 1)

1. A source file **F** is selected to be sent over the network at recipient node. Size **S** of this file (in Kilo Bytes) is saved at location **L** . For example if file size 102400 kilo byte (100 Mega Bytes) then S=1002400.
2. File **BF** (Binary File) is created from File **F** in memory buffer**T**. File BF Contains Binary information of source File **F**.
3. In File **BF** Initially each pattern of 8 bits is checked against other patterns available. This checking can be done using preexisting Master key files (**MKS**). It will help in faster key recognition. Secure files (files containing sensitive data like army information etc.) might need re-computation for each keys for every network transaction. It will increase security but it will be slower as compared to computation using preexisting Master key file (**MKS**).
4. Keys from the pool **KP** assigned to common patterns in file BF in first pass and File **BF1** is created in temporary memory.
5. File **BF** is deleted from the temporary memory.
6. Assigned keys in **BF1** are stored in file **KS**.
7. Once all keys are assigned, common patterns are searched again for second pass in newly created file **BF1**.
8. In second pass new keys are assigned to common patterns found in file **BF1** and File **BF2** is created in temporary memory.
9. File **BF1** is deleted from temporary memory.
10. Assigned keys in **BF2** are stored in file **KS**.
11. In third pass new keys are assigned to common patterns found in file **BF2** and File **BF3** is created in temporary memory.
12. File **BF2** is deleted from temporary memory.
13. Assigned keys in **BF3** are stored in file **KS**.
14. This pass process continues upto creation of **BF_n** File. Where **n** is the number of passes which are required to assign keys to source file F.
15. Final **KS** file is sent to recipient node.

Order for Compression	Binary pattern	Key assigned	Order for Decompression
1	00001 100	A	n
2	00101 100	B	n-1
3	11001 100	C	n-2
.	.	.	.
.	.	.	.
n-2	1AA1	X	3
n-1	1AB2	Y	2
n	1C12	Z	1

Table1: Final Ks file (these keys are shown in random order in the table)

16. File **F** is spitted into multiple subparts **Fsp**. where **Fsp={fsp1, fsp2, fsp3.…fspn}**.
17. Each subpart **Fsp={fsp1, fsp2, fsp3.…fspn}** is compressed into **Fspc** using File **KS**. Where **Fspc={fspc1, fspc2, fspc3,… Fspcn}.**
18. Each subpart**Fspc** is again compressed into **Fspcc** using any generic compression algorithm or any third party application.
    Where **Fspcc={fspcc1, fspcc2, fspcc3,… Fspccn}**.
19. This third party application should be same which is being used at receiver side. Otherwise generic compression algorithm is best choice to use at both (sender, receiver) end to avoid any kind of error.
20. Each **Fspcc={fspcc1, fspcc2, fspcc3,… Fspccn}** sent to receiver end using transmission control protocol/Internet protocol over the network.

Processes at receiver side (Phase 2)

1. Each subpart**Fspcc** is again decompressed into **Fspc** using generic decompression algorithm or same third party application which was used by sender. Here

   Fspcc={fspcc1, fspcc2, fspcc3,… Fspccn}

   And **Fspc={fspc1, fspc2, fspc3,… Fspcn}**.

2. This third party application should be same which is being used at receiver node. Otherwise always use generic compression algorithm to avoid any kind of error.

Retrieval Number: H10430688S319/19©BEIESP

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

3. Each subpart **Fspc={fspc1, fspc2, fspc3,… Fspcn}** is decompressed into **Fsp** using File **KS** for decompression. Where **Fsp={fsp1, fsp2, fsp3,… Fspn}.** For this decompression key file **KS** is scanned using bottom-up approach.
4. Each subpart **Fsp(Fsp1, Fsp2, Fsp... And so on)** is rejoined to produce original source file **F**.

Initial requirements.

Key pool KP is created using uniform patterns for binary codes [8]. KP act as data dictionary for regular patterns found in the binary file BF. For each irregular symbol or pattern found in binary file BF , keypool leave it as such until in any tth pass it find any uniform pattern. This uniform pattern might have been generated while converting binary digits into key codes in t-1th pass.

Calculation for number of sub parts.

Number of subparts are calculated using formula .

P=S/ps.

Where **P** is number of parts. **S** is size of source file **F**. ps is required size for each of the sub part of original file **F**.For example if size of file F is 1002400 Kilo Bytes andrequired **ps** is of size 100 Kilo Bytes then number of parts **P** is equal to (P=1002400/100=10024) 10024.

Hardware and software requirements

Central Processing Unit : Intel(R) Core(TM) i3-7100 CPU @ 3.90GHz

Random Access Memory :8 giga byte

Operating System : Windows 7 (64 bit).

Programming Framework : .Net 4.5.2

Language Used : Vb.Net

File sizes : An AVI (audio video interleaved) file of size 403 kilo byte (Up-scaled version on Clock.avi which is available with windows operating system) has been used to test this algorithm.

RESULT ANALYSIS

Compression Software	Initial File Size in Kilo Byte	Compression in size without algorithm	Compression In size with algorithm	% decrease in size
WinZip	403 KB	41	33	20.3
Winrar	403 KB	35	27	21.5
7-Zip	403 KB	27	22	19.8

Table 2 : Compression result in size

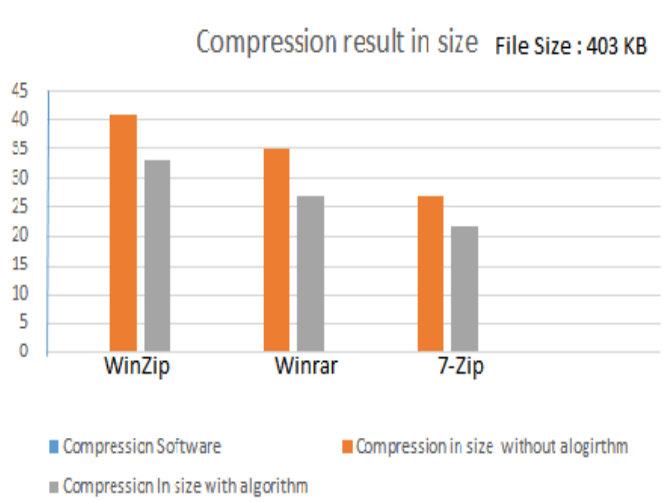

Chart 1: Compression result in size

It can be easily deduced that after using this algorithm compression ratio increased approximately by 20.5 % ([20.3+21.5+19.8]/3=20.5). Chart 1 and table 2clearly shows that this algorithm can improve applications of the existing algorithms considerably [9]. It will also help in improving media streaming as more data can be packed in a packet for same amount of bytes. All these tests has been done under normal compression level for all the third party applications [10].

Compression Software	Initial File Size in Kilo Bytes	Compression time taken without algorithm	Compression time taken with algorithm	% increase in time taken
WinZip	403	187 ms	190 ms	1.60%
Winrar	403	189 ms	192 ms	1.59%
7-Zip	403	191 ms	195 ms	2.09%

Table 3 : Compression result in time.

Retrieval Number: H10430688S319/19©BEIESP
Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

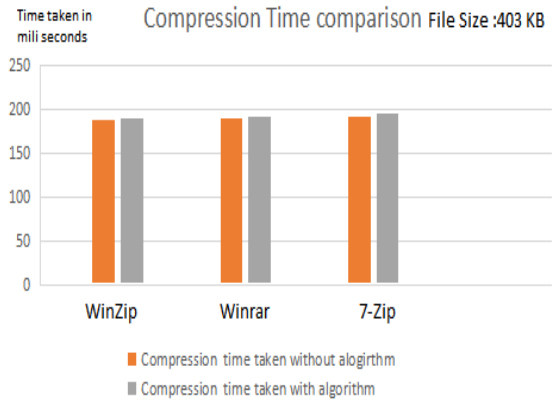

Chart 2: Compression result in time

Table 3 shows increase in time in milliseconds for using this algorithm as an over the top service for third party compression algorithms [11]. There is an increase of 3.33 milliseconds on an average ({[190-187]+[192-189]+[195-191]}/3=3.33 ms). It can be verified from chart 2 as well. In terms of performance ration this increase comes marginally around 1.76%([{[190-187]/[187*100]}+{[192-189]/[189*100]}+{[195-191] /[191*100]}] /3=1.76). It can be easily deduced from this observation that this performance overhead can easily be ignored for the benefits which are yielded by this algorithm.

Application of algorithm

This algorithm can be helpful in transferring data over wireless network [12]. This algorithm can send more data over the same bandwidth. It will not only help in increasing data transfer but also it will help in reducing network congestion.

As this algorithm can be used as an over the top application on each and every existing compression algorithm, it can be used with almost every kind of existing compression algorithm [13][14][15]. It will increase their efficiency without adversely affecting their real time performance [16].

Streaming media is booming now a days. There are so many Internet protocol television service providers for example Giga IPTV (Internet protocol television) , Gen IPTV etc [17][18]. These kind of services will be immensely benefited by this algorithm as it can help them to pack more channels in same bandwidth [19].

This algorithm can also be used for secure data transfer between sender and receiver . As data transfer over the network is already encrypted with keys in the file KS, middle man attack will be futile without having access to key file [20].

CONCLUSION

This algorithm has been tested along with various other third party compression applications. It has proved its mettle by improving their compression ratio without decreasing their real time performance. It can be safely assumed that this algorithm can be utilized for commercial purpose in various IT (information technology) applications. As kaizen is another name for continuous improvement, this algorithm can be optimized in future so that it could be adopted for all kind of data intensive applications.

REFERENCES

1. Ignacio Capurro et. el, "Efficient Sequential Compression of Multichannel Biomedical Signals", IEEE, pp 914-916, 2016
2. Weigang Li,Yu Yao, "Accelerate Data Compression in File System",IEEE, pp 615 - 615, 2016
3. Cornel Constantinescu ; Gero Schmidt, "Fast and Efficient Compression of Next Generation Sequencing Data", IEEE, pp 402-402, 2018
4. Jin Zhou, Chiman Kwan, "A Hybrid Approach for Wind Tunnel Data Compression", IEEE, pp 435-435, 2018
5. Weigang Li, "Optimize Genomics Data Compression with Hardware Accelerator", IEEE, pp 446-446, 2017
6. Jie Chen et. el, "OCT: A Novel Opportunistic Compression and Transmission Approach for Private Car Trajectory Data", IEEE, pp 401-401, 2018
7. Ping Wang et. el, "A TR069 WAN management protocol for WIA-PA Wireless sensor Networks", IEEE, pp 1-4, 2016
8. Sreekrishna Pandi et. el, "Reliable low latency wireless mesh networks — From Myth to reality", IEEE, pp 1-2, 2018
9. G. Kalfas et. el, "Network planning for 802.11ad and MT-MAC 60 GHz fiber-wireless gigabit wireless local area networks over passive optical networks", IEEE, pp 206-220, 2016
10. Sven Zehl et. el, "Hotspot slicer: Slicing virtualized home Wi-Fi networks for air-time guarantee and traffic isolation", IEEE, pp 1-3, 2017
11. Mohamed Labraoui et. el, Opportunistic SDN-controlled wireless mesh network for mobile traffic offloading",IEEE, pp 1-7, 2017
12. Bhaskar Prasad Rimai et. el, " Mobile data offloading in FiWi enhanced LTE-A heterogeneous networks",IEEE, pp 601-615, 2017
13. Mate Akos TUNDIK et.el, " Access-independent Cloud-based Real-Time Translation Service for Voice Calls in MobileNetworks ",IEEE, pp 1-6, 2018
14. Van-Giang Nguyen et. el, " SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey ",IEEE, pp 1567-1602, 2017
15. Ricardo Martínez et. el, " Integrated SDN/NFV orchestration for the dynamic deployment of mobile virtual backhaul networks over a multilayer (packet/optical) aggregation infrastructure",IEEE, pp A135-A142, 2017
16. Dawson Ladislaus Msongaleli, Kerem Kucuk " Reliability and cost-aware network upgrade for the next generation mobile networks ",IEEE, pp 496-501, 2017
17. Siyu Zhou et. el, " Low-latency high-efficiency mobile fronthaul with TDM-PON (mobile-PON) ",IEEE, pp A20-A26, 2018
18. Minseok Jang et. el, " A mobile ad hoc cloud for automated video surveillance system ",IEEE, pp 1001-1005, 2017
19. Toshikazu Terami et. el, " Evaluation of Information Dissemination Scheme Using Autonomous Clustering and Epidemic Routing Considering Mobile Core Network Load in Wireless Networks ",IEEE, pp 260-270, 2017
20. Ahmed Mohammed Mikaei et. el, " Performance evaluation of XG-PON based mobile front-haul transport in cloud-RAN architecture ",IEEE, pp 984-994, 2017

Retrieval Number: H10430688S319/19©BEIESP

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

AUTHOR PROFILE

Shiv Preet Scholar at Lovely Professional University
He is Ph.D. Scholar at Lovely Professional University and Assistant Manager at Anand Concast Limited . He is having 14+ years of experience in IT solution development and management.
He has published three research papers in reputed journals. His core areas are wireless networking. mobile networking, Compression and cryptography.

Dr. Amandeep, Ph.D in Computer science.
She is Associate Professor and Deputy Dean at Lovely Professional University. She is having 12+ years of experience as a Teacher and Educational Administrator.
She is currently Head of the Department in School of Computer Applications. She is guiding 6 Ph.D. scholars in the area of network security and data analytics. She has been a member of review boards of various journals and conferences in India and abroad.Her area of interest is cryptography, Network Security and Data Science. She has published 22 research papers published in various journals and conferences.

Retrieval Number: H10430688S319/19©BEIESP
Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication