2015

Design and Implementation of Network Transfer Protocol for Big Genomic Data

Mohammed Aledhari
Western Michigan University, mohammed.aledhari@wmich.edu

Fahad Saeed
Western Michigan University, fahadsaeed11@gmail.com

Design and Implementation of Network Transfer Protocol for Big Genomic Data

Mohammed Aledhari
Department of Computer Science
Western Michigan University
Kalamazoo, Michigan, 49008
mohammed.aledhari@wmich.edu

Fahad Saeed*
Department of Electrical and Computer Engineering
Western Michigan University
Kalamazoo, Michigan, 49008
fahad.saeed@wmich.edu

***Abstract*—Genomic data is growing exponentially due to next generation sequencing technologies (NGS) and their ability to produce massive amounts of data in a short time. NGS technologies generate big genomic data that needs to be exchanged between different locations efficiently and reliably. The current network transfer protocols rely on Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) protocols, ignoring data size and type. Universal application layer protocols such as HTTP are designed for wide variety of data types and are not particularly efficient for genomic data. Therefore, we present a new data-aware transfer protocol for genomic-data that increases network throughput and reduces latency, called Genomic Text Transfer Protocol (GTTP). In this paper, we design and implement a new network transfer protocol for big genomic DNA dataset that relies on the Hypertext Transfer Protocol (HTTP). Modification to content-encoding of HTTP has been done that would transfer big genomic DNA datasets using machine-to-machine (M2M) and client(s)-server topologies. Our results show that our modification to HTTP reduces the transmitted data by 75% of original data and still be able to regenerate the data at the client side for bioinformatics analysis. Consequently, the transfer of data using GTTP is shown to be much faster (about 8 times faster than HTTP) when compared with regular HTTP.**

I. Introduction

DNA molecules are made of two twisting paired strands, often referred to as a double helix. Each DNA strand is made of four chemical units, called nucleotide bases, which comprise the genetic "alphabet." The bases are adenine (A), thymine (T), guanine (G), and cytosine (C) [1]. NGS technologies generate a large amount of data and produce up to a petabyte of data in a single run [2] [3]. Many experimental biologists rely on cloud infrastructures and services to exchange, process, and analyze those datasets. However, most of the data migration takes place using traditional data-oblivious networking protocols. Since the amount of data is large it generally takes significant amount of time for the scientists to transfer the data over networks via the internet. We assert that networking protocols that are data-aware are essential to transfer large amount of genomic data efficiently. Since the cost of producing genomic data is decreasing dramatically, we expect experimental labs to produce even larger genomic data sets than currently possible.

*Correspondence should be addressed to Fahad Saeed at fahad.saeed@wmich.edu

As shown in table 1 [4] the capability of producing data is increasing and the cost of producing such data is decreasing rapidly. New technologies have increased Internet usage tremendously that exceed current bandwidth use, which leads to traffic congestion and other associated problems. There are multiple transfer protocols in TCP/IP application layer which work for all data types such as HTTP and File transfer protocol (FTP) [5]. Those protocols work over a secure protocol (TCP) that run within second (transport) layer [6] unlike UDP which is vulnerable [7]. Despite the fact that HTTP is the best choice for exchanging different data kinds between clients and servers [8], all of these protocols are data-oblivious i.e. they are not aware of the underlying data that is being transmitted. We assert that protocols that are data-aware, especially in the field of genomics, would be extremely useful for transference of data with high efficiency and throughput.

The paper begins with an author's contribution (section 2) and a literature review in section 3. Section 4 describes a background, and section 5 describes the proposed transfer protocol (HTTP content-encoding modification). Section 6 discuss the experiments and results, section 7 and 8 present a future work and conclusion respectively.

TABLE I
QUANTITATIVE ADVANCES SINCE THE HUMAN GENOME PROJECT(HGP)

	HGP		
Genome sequencing	1990	2003	2014
Generate cost	$1 Billion	$10-50 Million	3-5 thousand
Generate time	6-8 years	3-4 months	1-2 days

II. Author's contribution

Genomic data processing applications continue to grow in their scope, ambition and functionality. Therefore, we believe creating a data-aware transfer protocol would optimize each byte of transferred data, which increases network throughput, saves bandwidth, time and resources. This paper proposes a new network transfer protocol that relies on HTTP as the base protocol and modification are done suitable for big genomic data. Our paper assumes that the data consists of genomic data with only four base pairs (A, T, G, C) that need to be transferred, processed, visualized, and exchanged

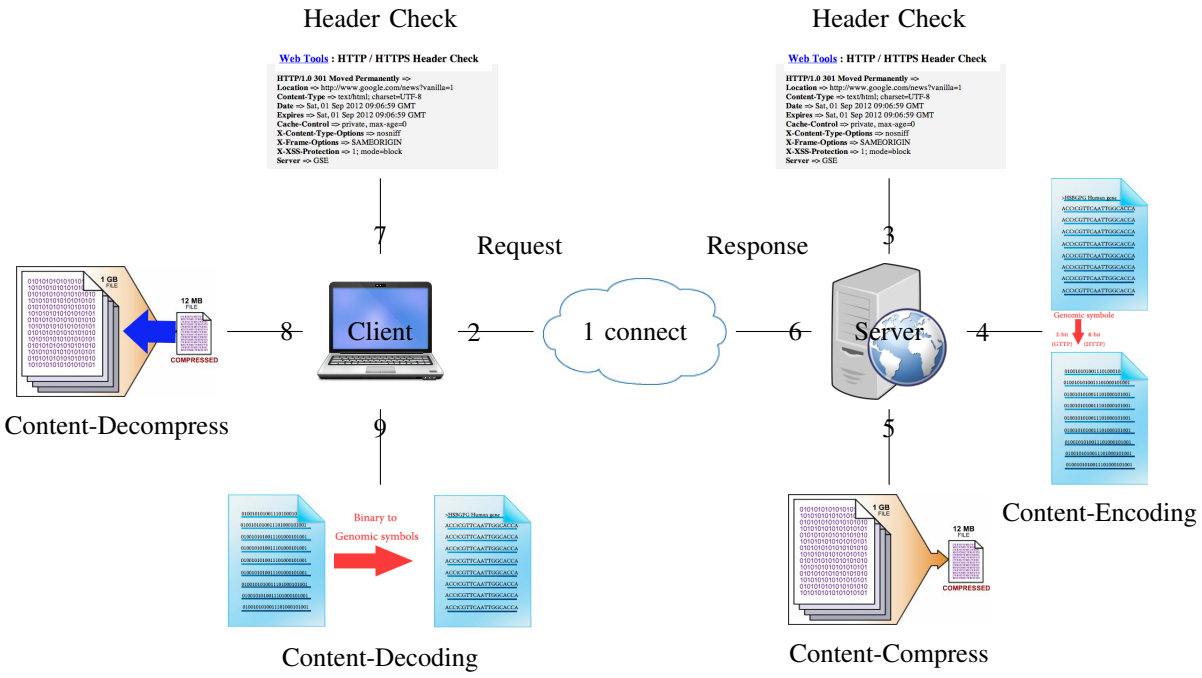

Fig. 1. GTTP architecture and life cycle. The life cycle consists of 9 stages like HTTP but the content-encoding is different i.e. 2-bits for each nucleotide in GTTP while 8-bit in HTTP.

over networks. Thus, modifying the HTTP content-encoding mechanism represents the key idea of this work since the HTTP contains many headers [9] that offer different functions for data transmission. Content-encoding is responsible for managing encoding and compressing algorithms and specifying the character set of transferred data. The simple form of HTTP is request-response mode after establishing a connection between client (browser) and server over TCP. In the proposed work four bytes are packed into one byte by assigning two bits for each genomic letter instead of eight bits as shown in table 2.

Our experiments suggest that the proposed encoding scheme

TABLE II
PROPOSED CODES TO USE IN GTTP CONTENT-ENCODING

	Genomic symbols			
Coding	A	T	G	C
HTTP	01000001	01010100	01000111	01000011
GTTP	00	01	10	11

reduces the amount of data that needs to be transferred by three fourths (3/4) since it converts every 32 bits to 8 bits. Moreover, proposed encoding approach decreases transfer time incredibly because less data needs to be transferred. After the customized encoding the encoded-data is passed into a compression algorithm such as GZIP to compress data [8]. Our results show that GTTP improves the network performance via minimizing latency and maximizing throughput. The GTTP architecture and timing sequence diagrams are presented in figures 1 and 2 respectively. In this paper, the performance of GTTP is evaluated in context of size and transfer time using a TCP as a baseline for performance evaluation.

III. Literature Review

We are not aware of any network transfer protocols specific for big genomic data. Several protocols are used to transfer genomic datasets such as HTTP, FTP, BitTorrent [10], GridFTP [11], GeneTorrent [12]. However, all of the protocols are data-oblivious and the genomic data are also transferred using same procedures as any other data. GridFTP relies on opening two connections one for control and another one for data itself. GridFTP allows third party to transfer files between two clients through separating control and data channels. However, for using GridFTP the client has to stay active at all times. In case the client state is lost the transferring must restart from beginning. BitTorrent and GeneTorrent increase their throughput by transferring data in parallel using multiple machines distributed on different locations. GeneTorrent is file transfer protocol which uses BitTorrent technique to transfer genomics data, which was originally designed to support distributed peer-to-peer (P2P) file transfer applications. In other words, GeneTorrent distributes same file(s) on different machines settled in different locations and configure those machines to transfer certain part(s) of that file(s) to a requester. Although higher throughput can be achieved by using multiple machines for transferring data, the underlying data is still transferred using general-purpose protocols. Therefore, there is a need to create a data-aware network transfer protocol for the DNA genomic data that uses minimum resources of network to deliver data efficiently.

In this paper we modify the HTTP protocol that is specific

to genomics datasets and can transfer data using different network topologies (one-to-one, one-to-many, and many-to-many). Specifically we modified the content-encoding of the HTTP protocol specific to genomic data which allows us to transfer much more data than is possible using generic encoding. The focus of this paper is to design and develop a data-aware networking protocol, which is faster, efficient and lightweight as compared to the existing protocols that are data-oblivious. This paper assumes that the data that needs to be transferred comprises of DNA sequences with four nucleotides (A, T, G, C). This paper introduces a new network transfer protocol dedicated for genomic data that relies on HTTP with modification to encoding mechanism that increase network performance and throughput.

IV. Background

A. HTTP overview

The HTTP is an application layer protocol of TCP/IP model which allows transfer of data over the Internet. HTTP works in request-response mode by sending a request for certain data from the client (requester) to the server (data source). The request contains extra information such as method (GET, PUT, POST, etc.), a uniform resource locator (URL), and headers. A complete cycle of transportation in HTTP starts by establishing a client-server connection using 3-way TCP handshaking. After that, client sends a request with some headers using supported methods. Server receives the request and checks headers to determine the needed contents via an accepted data type and content-encoding. The server encodes (here our contribution), compresses, and transmits required data over the network medium as seen in figure 1.

B. Standard HTTP Headers

Content-encoding header is one of HTTP headers that specify the range and the kind of data to be transferred, the compression algorithm [9] and character set. The charset names may be up to 40 characters taken from the printable characters of US-ASCII [13]. Transfer-encoded indicates transferred data size, either known (static) or unknown (dynamic). Transfer-encoded header assigns fixed value if data is stored or chunked value if generate on the fly. Genomic data has a limited character size (4-alphbets) and we exploit this in our content-encoding for greater efficiency and throughput.

C. Compress Data in Websites Using GZIP

GNU zip (GZIP) is utility relies on the DEFLATE compression algorithm [14] that mix LZ77 [15] [16] and Huffman coding [17] together. LZ77 algorithm works by replacing repeated strings with references. Each reference has two values, distance to last seen and string length. Distance is limited to 32KB and length is limited to 258B. Huffman coding is a variable-length coding that works by assigning shorter codes for the more repeated characters. A variable-length coding needs to know the start and end points for each character to decode it. Huffman coding solves this problem by creating a prefix code, where no codeword is prefix of another one. Therefore, GZIP relies on three parameters, which are distance, string length, and beginning and ending of each character. Based on [18] GZIP and DEFLATE compress the data to reach 25% of original size. GZIP has the added advantage that most browsers around the world support it.

V. Proposed Transfer Protocol

The GTTP determines location of required data to encode and compress to transfer it over the network. The main contribution begins when server starts transferring needed data by encoding into 2-bits form instead of 8-bits. GTTP reads each 4 letters and store them in only one letter to reduce data by 75%. Generated encoded data (25% of original) is then passed into compression algorithm (gzip) which is part of the HTTP and reduces data by 75%. The binary encoded form for each genomic letter can be found in table 2. For instance, GTTP encoding will encode genomic ASCII of AAAA into 00000000 binary form and save 24-bits to pass 8-bits into compress stage that reduce data by 75%. The total of transmitted data would be 6.25% of original data or less (25% from encoded stage * 25% from compress stage) as shown in equations 1 and 2 and table 2. Let us assume we have genomics file *Gen* = $\{a_1, a_2, a_3, ..., a_n\}$ $a_i \in$ {A, T, G, C} Where n is the number of characters in the file and each character will encoded into $c(ai)$ as $c(ai)$ = {A = 00, T = 01, G = 10, C = 11} = 2 bit From that we create a new equation as in

$$B(Gen) = \sum_{i=1}^{n} a_i length(c_i) \tag{1}$$

Where *B(Gen)* is binary convert of *Gen* file, and $length(c_i)$ represents the length of coded a_i in bits. For example, suppose we want to send a FASTA file (genomics) format that contains 100,000 characters, then we would get in:

1) Normal coding would require 100,000 * 8 = 800,000 bits.
2) Proposed encoding uses only 100,000 * 2 = 200,000 bits with 75% saving.

Therefore, we obtain on a new equation as in

$$B(Gen) = \frac{1}{4} \sum_{i=1}^{n} a_i length(c_i) \tag{2}$$

$$Computation_{cost}(C_c) = O\left(HC_c + CE_c + CC_c \right) \tag{3}$$

$$Communication_{cost}(E_c) = O\left(3ways_c + transfer_c \right) \tag{4}$$

$$Total_{cost}(T_c) = (C_c) + (E_c) \tag{5}$$

Equations 3-5 show the total cost for transceiver data over HTTP and GTTP. Where HC_c refers to header check cost, CE_c refers to content-encoding {here is our contribution}, CC_c indicates to content-compress, $3ways_c$ refers to 3-ways TCP handshaking, and transfer represents network bandwidth. It is easy to conclude minimizing content-encoding leads to minimize data amount to be compress and then reduce

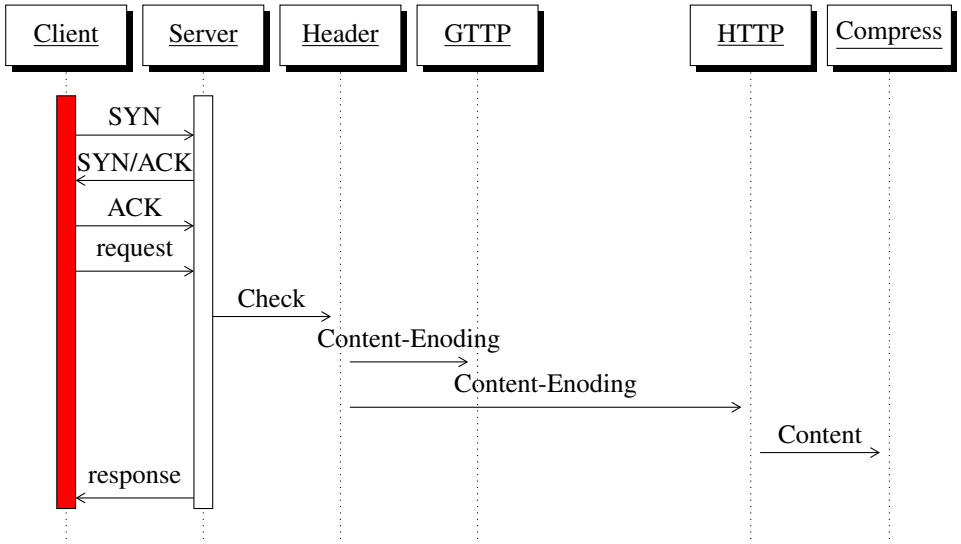

Fig. 2. Time sequence diagram for HTTP and GTTP life cycle. Client requests genomic DNA datasets from a server after completing 3-way handshaking connection. The server starts by checking headers then pass genomic nucleotides to binary converting stage. After that binary data is passed to compression stage to transfer through connection media

transferred data. So that minimizing CE_c significantly impacts on T_c.

To get 800,000 * 0.25 = 200,000 bits in normal way whereas 200,000 * 0.25 = 50,000 bits in the propose approach. Consequently, we will end up with sending 0.0625 of original data, which is much more efficient than existing techniques.

GTTP protocol can be summarized in algorithm 1.

Briefly, in the first stage produces only 25% of original data and the second stage will produce (25%) of first stage. Thus, GTTP will send about (6.25%) of original data that causes a significant increase in the network performance. It is hypothesized that the proposed study will minimize impairment in transmitting genomic big data using GTTP as well as improving network throughput by sending less data. GTTP is first data-aware protocol for genomic data that modifies the HTTP encoding mechanism to increase data transfer rates at higher throughput using same network bandwidth and hardware.

Algorithm 1 Proposed GTTP content-encoding

```
1: procedure ENCODING
2:     if inputStream.hasGenomicFileheader then
3:         outputStream.write(GenomicFileheader)
4:     end if
5:     while !inputStream.EOF do
6:         genomicChar ← inputStream.GetChar().
7:         twoBits ← towBitsEncoding(genomicChar).
8:         oneByteStore.store(twoBits).
9:         if oneByteStore.ISFull() then
10:            outputStream.write(oneByteStore).
11:            oneByteStore.empty().
12:        end if
13:    end while
14:    if !oneByteStore.ISEmpty() then
15:        outputStream.write(oneByteStore).
16:        outputStream.write(NumOfExtraBits).
17:    end if
18: end procedure
```

VI. Experiments and Results

In order to validate our theoretical results we performed experiments on real (table 3) as well as simulated (table 4) genomic data sets up to 50GB. For our experiments we implemented a genomic data simulator that generates genomic DNA datasets randomly in FASTA format and works in two modes.

- Auto Genomic DNA Generator.
- Manual Genomic DNA Generator.

In auto-mode, the application generates genomic dataset of specific size with random frequency of DNA nucleotides in the data. In the case of manual mode, we are able to control the frequency of genomic DNA symbols (A,T,G,C) in the data. In both modes, we can specify the size of the file to be generated (in MB or GB). For instance, 10GB-A-35 means generates genomic DNA dataset with size of 10GB with occurrence rate (35%) for symbol (A) and the remaining (65%) for three other genomic symbols are randomly chosen. To validate our experiments, we generated 18 genomic DNA datasets between 100MB-50GB with 3 percentages (25%, 35%, and 50%) for the symbol (A). We used different rates to assure that our strategy is independent of nucleotide frequencies in the data. The genomic DNA generator algorithm can be seen in algorithm 2. We used (FASTA format) as genomic files to exchange between two machines (client-server). Experimental results can be seen in figures 3-10. Those results came from our implementation using visual C# language version (2013)

TABLE III
REAL GENOMIC DNA DATASETS USED FOR EXPERIMENTS

Sequence ID	Organism	Platform	Size(MB)	Renamed
gi|157704448|ref|AC_000133.1|	Homo sapiens chromosome 1	ILLUMINA	9.44	1
gi|157731950|ref|AC_000135.1|	Homo sapiens chromosome 3	ILLUMINA	9.73	2
gi|157718668|ref|AC_000151.1|	Homo sapiens chromosome 19	ILLUMINA	12.38	3
gi|157713538|ref|AC_000149.1|	Homo sapiens chromosome 17	ILLUMINA	12.67	4
gi|528476670|ref|NC_018912.2|	Homo sapiens chromosome 1	ILLUMINA	12.76	5
gi|528476558|ref|NC_018928.2|	Homo sapiens chromosome 17	ILLUMINA	13.85	6
gi|157734152|ref|AC_000138.1|	Homo sapiens chromosome 6	ILLUMINA	15.02	7
gi|528476658|ref|NC_018914.2|	Homo sapiens chromosome 3	ILLUMINA	15.61	8
gi|157704449|ref|AC_000142.1|	Homo sapiens chromosome 10	ILLUMINA	16.20	9
gi|157704453|ref|AC_000144.1|	Homo sapiens chromosome 12	ILLUMINA	17.96	10
gi|528476637|ref|NC_018917.2|	Homo sapiens chromosome 6	ILLUMINA	17.97	11
gi|157734237|ref|AC_000155.1|	Homo sapiens chromosome X	ILLUMINA	18.55	12
gi|528476628|ref|NC_018918.2|	Homo sapiens chromosome 7	ILLUMINA	18.56	13
gi|528476665|ref|NC_018913.2|	Homo sapiens chromosome 2	ILLUMINA	18.84	14
gi|157734151|ref|AC_000137.1|	Homo sapiens chromosome 5	ILLUMINA	19.13	15
gi|528476546|ref|NC_018930.2|	Homo sapiens chromosome 19	ILLUMINA	22.08	16
gi|157704452|ref|AC_000143.1|	Homo sapiens chromosome 11	ILLUMINA	24.13	17
gi|74230042|gb|CH471078.2|	Homo sapiens 211000035833619 genomic scaffold	ILLUMINA	31.97	18
gi|71517006|gb|CH471076.1|	Homo sapiens 211000035835228 genomic scaffold	ILLUMINA	32.26	19
gi|74230049|gb|CH471059.2|	Homo sapiens 211000035844098 genomic scaffold	ILLUMINA	55.86	20
gi|74273668|gb|CM000265.1|	Homo sapiens chromosome 14	ILLUMINA	84.46	21
gi|74422211|gb|CM000231.2|	Rattus norvegicus chromosome 1	ILLUMINA	84.97	22
gi|74230054|gb|CH471051.2|	Homo sapiens 181000117649897 genomic scaffold	ILLUMINA	100.39	23
gi|74273673|gb|CM000260.1|	Homo sapiens chromosome 9	ILLUMINA	107.93	24
gi|74273671|gb|CM000262.1|	Homo sapiens chromosome 11	ILLUMINA	127.84	25
gi|74273659|gb|CM000274.1|	Homo sapiens chromosome X	ILLUMINA	150.32	26

TABLE IV
GENERATED GENOMIC DNA DATASETS USING OUR GENERATOR USED IN EXPERIMENTS

Datasets	Number of DNA Symbols in each Dataset							
	A	C	G	T	A%	C%	G%	T%
100M[25]	26843545	26843645	31808042	21878898	0.25%	0.25%	0.30%	0.20%
100M[35]	37580963	30868915	13425306	25498926	0.35%	0.28%	0.13%	0.24%
100M[50]	53687091	14786579	25675457	13224952	0.50%	0.14%	0.24%	0.12%
500M[25]	134217728	149129757	149112975	104410196	0.25%	0.28%	0.28%	0.19%
500M[35]	187904819	153550318	116324344	79091072	0.35%	0.29%	0.21%	0.15%
500M[50]	268435456	93792797	100263349	74378798	0.50%	0.17%	0.19%	0.14%
1G[25]	268435456	219637924	292809252	292858680	0.25%	0.21%	0.27%	0.27%
1G[35]	375809638	221793199	254370797	221767473	0.35%	0.20%	0.24%	0.21%
1G[50]	536870912	203878326	101922932	231068630	0.50%	0.19%	0.09%	0.22%
5G[25]	1342177280	1283854849	1517198501	1225475930	0.25%	0.24%	0.28%	0.23%
5G[35]	1879048192	1143192988	1203320398	1143143958	0.35%	0.21%	0.22%	0.22%
5G[50]	2684354560	723874757	1176284699	784189984	0.50%	0.13%	0.22%	0.15%
10G[25]	2684354560	2467876125	2338051024	3247131411	0.25%	0.23%	0.22%	0.30%
10G[35]	3758096384	2594087000	2532325440	1852902248	0.35%	0.24%	0.24%	0.17%
10G[50]	5368709120	894831061	2013252324	2460615495	0.50%	0.08%	0.19%	0.23%
50G[25]	13421772800	13421823505	14380428192	12463041103	0.25%	0.25%	0.27%	0.23%
50G[35]	18790481920	10550108277	9738567871	14607897292	0.35%	0.20%	0.18%	0.27%
50G[50]	26843545600	10949303940	7064179103	8830011357	0.50%	0.20%	0.14%	0.16%

over two different environments (machine specifications and bandwidth) as following:

- `1-26 (real) genomic DNA datasets`
  Windows 8.1 pro (64-bit), Intel Core i7 with clock speed of 2.4 GHz. The system is equipped with 8GB RAM, L2 Cache (per Core) of 256 KB and L3 Cache of 6 MB over 37.59 Mb/s DOWNLOAD and 4.22 Mb/s UPLOAD speeds.
- `100MB[xx]-50GB[xx] (generated using our genomic generator) genomic DNA datasets`
  Windows 8.1 Enterprise (64-bit), Intel Xeon CPU W3565 with clock speed of 3.20GHz, 4 Cores, 8 logical processors. The system equipped with 24GB over 76.59 Mb/s DOWNLOAD and 103 Mb/s UPLOAD speeds.

First 26 tested files came from The National Center for Biotechnology Information advances science and health (NCBI) [19] and remaining 100MB[xx]-50GB[xx] we generated them randomly using generator for FASTA files implemented by us as showed in algorithm 2.

Algorithm 2 Genomic DNA Generator Algorithm

```
1: procedure GENERATE
2:     FileSize ← Get Genomic File Size .
3:     TotalSymoblsToBeGenerated ← FileSize .
4:     Mode ← Get Generation Mode.
5:     if Mode = Manually then
6:         Symbol ← Get Genomic Symbol to control its ← frequency.
7:         Ratio ← Get the Genomic Symbol's percentage .
8:         SymbolGenerate(OutputFile,Symbol, ← Ratio * FileSize).
9:         TotalSymoblsToBeGenerated ← FileSize *(1-Ratio).
10:    end if
11:    RandomlyInitArray(SymbolsArray). ▷ SymbolsArray will be filled with genomic symbols randomly, the array size is calculated randomly too
12:    while TotalSymoblsToBeGenerated > 0 do
13:        ArrayIndex ← Random.GetNumber % ← SymbolsArray.Length. ▷ Pick an element from SymbolsArray randomly
14:        GenomicSybmol ← SymbolsArray[ArrayIndex].
15:        OutputFile.write(GenomicSybmol).
16:        TotalSymoblsToBeGenerated ← TotalSymoblsToBeGenerated -1 .
17:    end while
18: end procedure
```

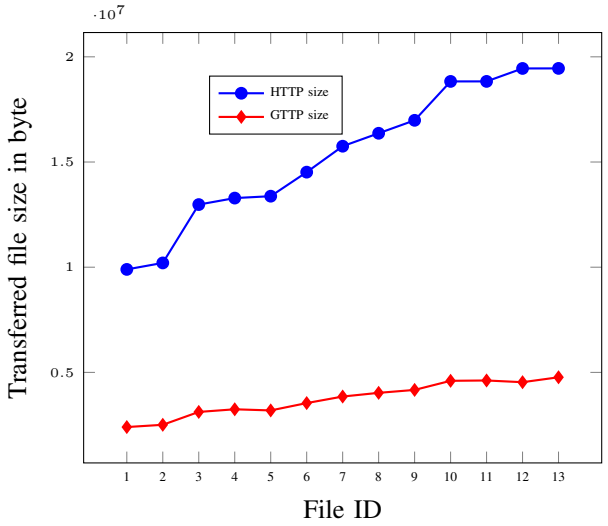

Fig. 3. Transfer size of first actual 13 genomic DNA datasets (9.44MB-18.56MB) over both HTTP and GTTP protocols with range of 37.59 Mb/s DOWNLOAD and 4.22 Mb/s UPLOAD speeds.

Our experiments start by client requesting a specific genomic data file from the server. The server encodes required data into binary form using our GTTP protocol and passes it to gzip algorithm to send zipped file to the client. When a client receives a zipped file, it decompresses and decodes it to get

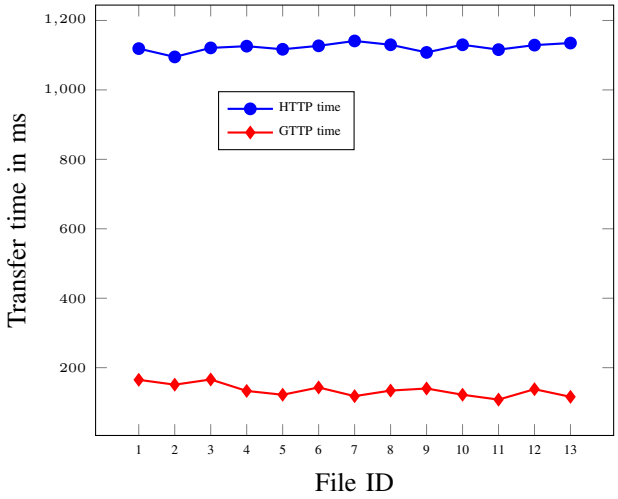

Fig. 4. Transfer time of first actual 13 genomic DNA datasets (9.44MB-18.56MB) over both HTTP and GTTP protocols with range of 37.59 Mb/s DOWNLOAD and 4.22 Mb/s UPLOAD speeds.

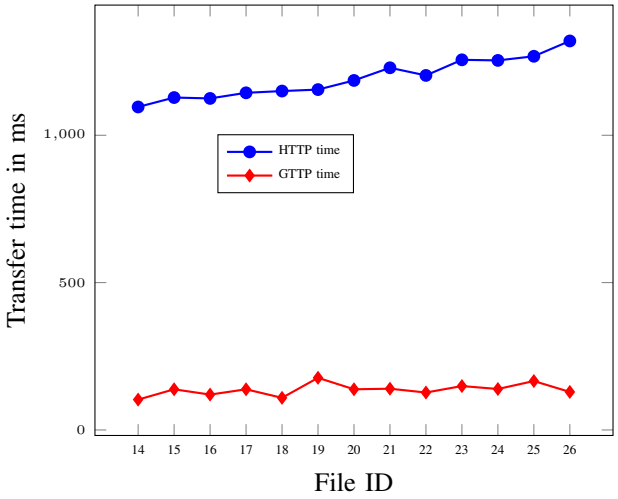

Fig. 6. Transfer time of second actual 13 genomic DNA datasets (18.84MB-150.32MB) over both HTTP and GTTP protocols with range of 37.59 Mb/s DOWNLOAD and 4.22 Mb/s UPLOAD speeds.

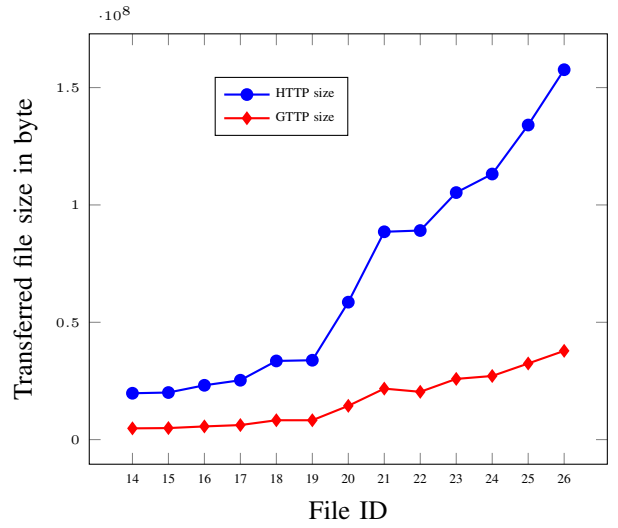

Fig. 5. Transfer size of second actual 13 genomic DNA datasets (18.84MB-150.32MB) over both HTTP and GTTP protocols with range of 37.59 Mb/s DOWNLOAD and 4.22 Mb/s UPLOAD speeds.

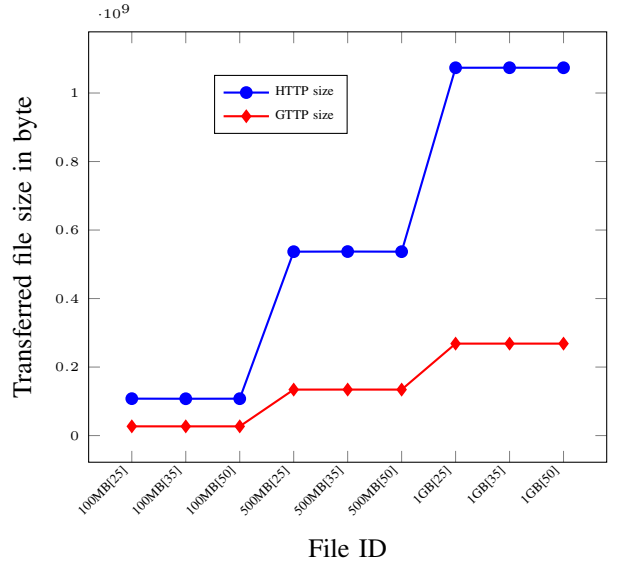

Fig. 7. Transfer size of first simulated 9 big genomic DNA datasets (100MB[xx]-1GB[xx]) over both HTTP and GTTP protocols with range of 76.59 Mb/s DOWNLOAD and 103 Mb/s UPLOAD speeds.

original genomic data. We also implemented a normal HTTP request-response to compare and assess with our GTTP results. Experiments were performed on different genomic files to determine the minimum time required to transfer them using both protocols, HTTP and GTTP. Our experiments show that GTTP encoded mechanism decreased transmitted data, which also reduced the transfer time. Figures 3, 5, 7, and 9 show transferring *data size* via both protocols HTTP and GTTP. For example, experiment 2 transfers 10203125 bytes using HTTP, while we only need to transfer 2508540 bytes of data using GTTP, which is 75% decrease in the data set size. Also, experiment 33 (1GB[25]) transfers 1073741312 bytes (1GB) through HTTP, whereas 268435328 bytes (0.25GB) through GTTP with saving 0.75 to send 0.25 instead. Figures 4, 6, 8, and 10 show *transferring time* for genomic files using both protocols HTTP and GTTP. For instance, HTTP needed 1095 ms to transfer 10203125 bytes from experiment 2, whereas GTTP spent only 151 ms to transfer the same file with 7.27 times faster. Also, HTTP spent 1880973 ms to transfer 10737413120 bytes (10GB[25]) from experiment 39, while GTTP spent only 208439 ms to transfer the same file, which is 9.02 times faster.

Our proposed strategy and the experimental results show that using our data-aware strategy significant reduction in size and increase in throughput can be achieved when transferring genomic data sets. The results are also summarized in table 5. To our knowledge this is first data-aware protocol for genomic

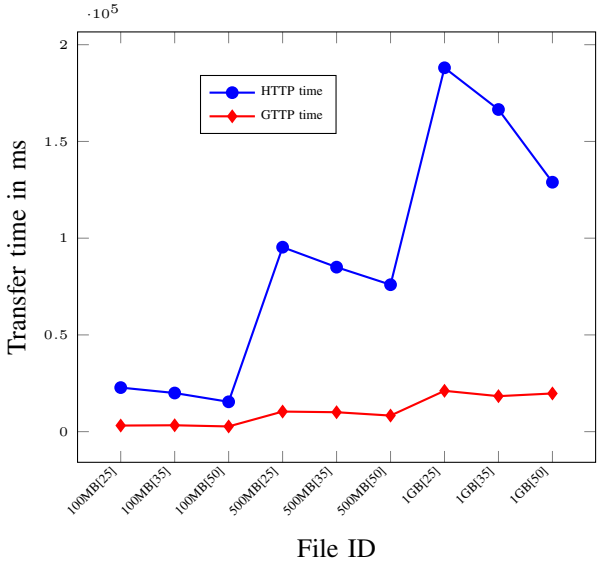

Fig. 8. Transfer time of first simulated 9 big genomic DNA datasets (100MB[xx]-1GB[xx]) over both HTTP and GTTP protocols with range of 76.59 Mb/s DOWNLOAD and 103 Mb/s UPLOAD speeds.

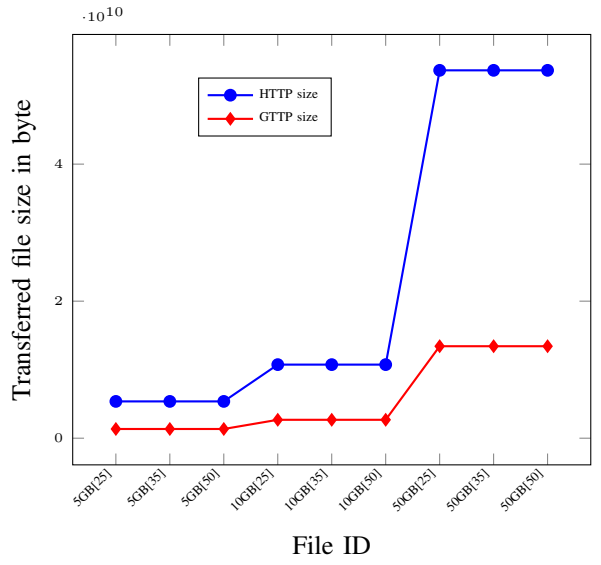

Fig. 9. Transfer size of second simulated 9 big genomic DNA datasets (5GB[xx]-50GB[xx]) over both HTTP and GTTP protocols with range of 76.59 Mb/s DOWNLOAD and 103 Mb/s UPLOAD speeds.

data.

VII. Future work

The proposed data-aware protocol is applied on a point-to-point network topology (single server-client). Future experiments will include using multiple servers to increase throughputs either using cluster nodes or BitTorrent technique. Currently, GTTP is designed for transfer of fixed file size. Future works include developing an online protocol that could transfer data from the NGS machines in real-time using our data-aware encoding strategies. Elimination of headers that are not used by our protocol will also be useful.

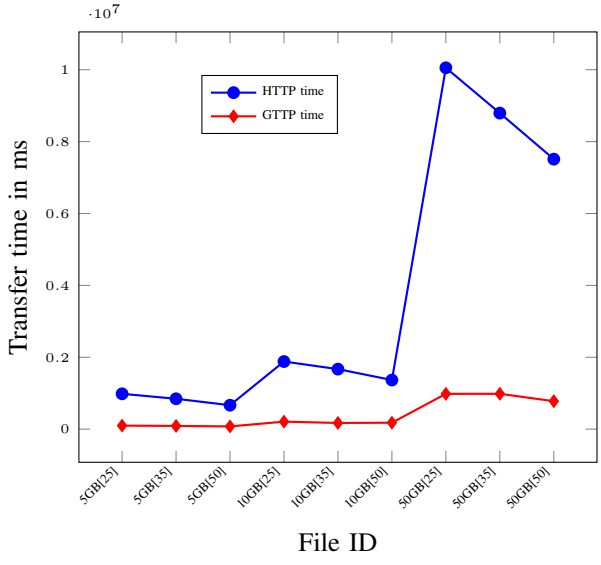

Fig. 10. Transfer time of second simulated 9 big genomic DNA datasets (5GB[xx]-50GB[xx]) over both HTTP and GTTP protocols with range of 76.59 Mb/s DOWNLOAD and 103 Mb/s UPLOAD speeds.

TABLE V
Optimization obtained by GTTP content-encoding for the transferred size and speed up time against HTTP

Datasets	size	time	Datasets	size	time
1	0.25x	6.79x	23	0.25x	8.42x
2	0.25x	7.27x	24	0.25x	9.01x
3	0.25x	6.77x	25	0.25x	7.64x
4	0.25x	8.50x	26	0.25x	10.24x
5	0.25x	9.16x	100MB[25]	0.25x	7.23x
6	0.25x	7.87x	100MB[35]	0.25x	6.06x
7	0.25x	9.66x	100MB[50]	0.25x	5.73x
8	0.25x	8.40x	500MB[25]	0.25x	9.19x
9	0.25x	7.94x	500MB[35]	0.25x	8.47x
10	0.25x	9.23x	500MB[50]	0.25x	9.14x
11	0.25x	10.34x	1GB[25]	0.25x	8.91x
12	0.25x	8.16x	1GB[35]	0.25x	9.08x
13	0.25x	9.80x	1GB[50]	0.25x	6.53x
14	0.25x	10.65x	5GB[25]	0.25x	10.05x
15	0.25x	8.20x	5GB[35]	0.25x	9.46x
16	0.25x	9.38x	5GB[50]	0.25x	8.80x
17	0.25x	8.29x	10GB[25]	0.25x	9.02x
18	0.25x	10.54x	10GB[35]	0.25x	9.70x
19	0.25x	6.53x	10GB[50]	0.25x	7.64x
20	0.25x	8.59x	50GB[25]	0.25x	10.24x
21	0.25x	8.78x	50GB[35]	0.25x	8.94x
22	0.25x	9.50x	50GB[50]	0.25x	9.66x

VIII. Conclusion

We presented a new strategy for network transfer protocol by designing and implementing a novel data-aware protocol for genomic sequences, called GTTP. Our results showed that the data transfer time between the 2 machines using our approach is much faster (about 8 times faster than HTTP

especially for big files) and more network throughput is achieved when compared to a regular HTTP. Such data-aware protocols will also be useful for other Big Data domains.

ACKNOWLEDGMENT

This work was supported in part by grants NSF CNS-1250264, NSF CNS-1441384 and NSF CCF-1464268. The authors would like to thank the members of the Parallel Computing and Data Science (PCDS) lab at Western Michigan University for their valuable input while in the process of developing this paper.

REFERENCES

[1] A. Lesk, *Introduction to genomics*. Oxford University Press, 2011.
[2] A. O'Driscoll, J. Daugelaite, and R. D. Sleator, "'big data', hadoop and cloud computing in genomics," *J Biomed Inform*, vol. 46, no. 5, pp. 774–81, Oct 2013.
[3] F. Saeed, A. Perez-Rathke, J. Gwarnicki, T. Berger-Wolf, and A. Khokhar, "A high performance multiple sequence alignment system for pyrosequencing reads from multiple reference genomes," *Journal of parallel and distributed computing*, vol. 72, no. 1, pp. 83–93, 2012.
[4] M. Bethesda, "Genomics on productive trajectory a decade after completing flagship project," *NHGRI celebrates 10th anniversary of the Human Genome Project*, April 2013.
[5] J. Postel and J. Reynolds, "File transfer protocol," 1985.
[6] J. Postel, "Transmission control protocol," 1981.
[7] Postel, "User datagram protocol," in *RFC 768, USC/Information Sciences Institute*, ISI, Ed. Citeseer, 1980.
[8] O. Oyman and S. Singh, "Quality of experience for http adaptive streaming services," *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 20–27, 2012.
[9] J. C. Mogul and M. Nottingham, "Http header field registrations," 2005.
[10] B. Cohen, "The bittorrent protocol specification," 2008.
[11] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The globus striped gridftp framework and server," in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, ser. SC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 54–. [Online]. Available: http://dx.doi.org/10.1109/SC.2005.72
[12] D. Maltbie, L. Ganeshalingam, and P. Allen, "System and method for secure, high-speed transfer of very large files," Jun. 24 2013, uS Patent App. 13/925,747.
[13] N. Freed and J. Postel, "Iana charset registration procedures," BCP 19, RFC 2978, October, Tech. Rep., 2000.
[14] L. P. Deutsch, "Deflate compressed data format specification version 1.3," 1996.
[15] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on information theory*, vol. 23, no. 3, pp. 337–343, 1977.
[16] A. Ziv, Jacob; Lempel, "Compression of individual sequences via variable-rate coding," *Information Theory, IEEE Transactions on*, vol. 24, no. 5, pp. 530–536, 1978.
[17] D. A. Huffman *et al.*, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.
[18] X. Chen, W. Wang, and G. Wei, "Impact of http compression on web response time in asymmetrical wireless network," in *NSWCTC'09: Proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2009, pp. 679–682.
[19] B. (MD), *The NCBI Handbook*, 2nd ed. National Center for Biotechnology Information (US), 2013. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK143764/