Models for Evaluating Effective Throughputs for File Transfers in Mobile Computing

Armen Dzhagaryan, Aleksandar Milenković
Electrical and Computer Engineering
The University of Alabama in Huntsville
Huntsville, AL

Abstract — The importance of optimizing data transfers between mobile computing devices and the cloud is increasing with an exponential growth of mobile data traffic. Lossless data compression can be essential in increasing communication throughput, reducing communication latency, achieving energy-efficient communication, and making effective use of available storage. In this paper we introduce analytical models for estimating effective throughputs of uncompressed and compressed data file transfers that utilize common compression utilities. The proposed analytical models are experimentally verified using modern smartphones as mobile devices. The proposed analytical models are instrumental in developing a framework for seamless optimization of file transfers in mobile computing.[1]

Index Terms — Mobile computing, Data compression, Performance Evaluation, Energy-aware systems

I. Introduction

Mobile computing devices such as smartphones, tablets, and e-readers have become the dominant platforms for consuming digital information. The data traffic originating on mobile computing devices and Internet-of-Things (IoT) platforms has been growing exponentially over the last several years. A report from Cisco states that the global mobile data traffic grew 69% in 2014 relative to 2013, reaching 2.5 exabytes per month, which is over 30 times greater than the total Internet traffic in 2000 [1]. It is forecast that the global mobile data traffic will grow nearly 10-fold from 2014 to 2019, reaching 24.3 exabytes per month.

Lossless data compression can increase communication throughput, reduce latency, save energy, and increase available storage. However, compression introduces additional computational overhead that may exceed any gains due to transferring or storing fewer bytes. Compression utilities on mobile computing platforms differ in compression ratio, compression and decompression speeds, and energy requirements. In addition, compression utilities support a range of compression levels, with lower levels favoring speed and higher levels favoring better compression ratio. When transferring data, we would like to have an agent to determine whether compressed transfers are beneficial, and, if so, select the most beneficial compression utility and compression level. A first step toward designing such an agent is to obtain a good understanding of various parameters impacting data transfers.

Lossless data compression is currently being used to reduce the required bandwidth during file downloads and to speed up web page loads in browsers. Google's Flywheel proxy [2], Google Chrome [3], Amazon Silk [4], as well as the mobile applications Onavo Extend [5] and Snappli [6] use proxy servers to provide HTTP compression for all pages during web browsing. For file downloads, several Google services, such as Gmail and Drive, provide *zip* compression [7] of files and attachments [8]. Similarly, application stores such as Google Play and Apple's App Store use *zip* or *zip*-derived containers for application distribution. Several Linux distributions are also using common compression utilities such as *gzip*, *bzip2*, and *xz* for their software repositories.

The importance of lossless compression in network data transfers has also been recognized in academia [9]–[13]. Recent studies [14], [15] focused on measurement-based experimental evaluation of compressed and uncompressed file transfers on the state-of-the-art mobile devices. These studies showed that selected compressed transfers over WLAN and cellular interfaces outperform corresponding uncompressed file transfers. However, not a single combination of a compression utility and a compression level performs the best for all file transfers and network conditions. A number of parameters may impact the effectiveness of file uploads and downloads initiated on a mobile device. These parameters include the type of network interface (e.g., cellular, WLAN), network connection throughput and latency, type and size of transferred files, and mobile device performance characteristics.

In this paper we introduce analytical models for estimating the effectiveness of uncompressed data transfers and compressed data transfers that use common compression utilities. As a measure of effectiveness, we use the effective upload and download throughputs expressed in megabytes per second. The analytical models describe effective upload and download throughputs for uncompressed and compressed network transfers as a function of parameters such as:

- Uncompressed (raw) file size;
- Local (de)compression throughput;
- Compression ratio; and

[1] This material is based upon work supported in part by the National Science Foundation under Grants No. 1205439 and 1217470. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

- Network parameters including network connection throughput and time to setup a network connection.

We experimentally verify the proposed models on Google's Nexus 4 and OnePlus One smartphones. The proposed models are instrumental in developing a framework for optimized data transfer between mobile computing devices and the cloud. The framework relies on agents running on mobile devices and the cloud to select effective modes of data upload and download transfers. For a given file, the framework will utilize the analytical models to estimate effectiveness of different file transfer options and to select the most effective approach.

The rest of this paper is organized as follows. Section II presents background and motivation for our study. It gives a system view of file transfers (II.A), describes target devices used in experimental verification of the models (II.B), and makes a case for optimizing data transfers (II.C). Section III describes the design and verification of analytical models for uncompressed file transfers. Section IV describes the design and verification of analytical models for compressed file transfers. Finally, Section V summarizes our findings and draws conclusions.

II. Background and Motivation

A. Data Transfer between Mobile Devices and the Cloud

Fig. 1 illustrates file uploads and downloads initiated from a mobile device. A data file can be uploaded uncompressed or compressed. In case of uncompressed uploads, an uncompressed file (UF) is uploaded over a network interface. In case of compressed uploads, the uncompressed file is first compressed locally on the device, and then a compressed file (CF) is uploaded over the network. Similarly, a file can be downloaded from the cloud uncompressed or compressed. In case of compressed downloads, a compressed version of the requested file is downloaded from the cloud, and then the compressed file is decompressed locally on the mobile device. Compressed uploads and downloads utilize one of the available compression utilities and one of the available compression levels.

In this paper we consider six common compression utilities listed in TABLE I for compressed file transfers. We have selected relatively fast *gzip* and *lzop* utilities, as well as *bzip2* and *xz*, which provide a high compression ratio. As many modern mobile devices include multicore processors, we also consider *pigz* and *pbzip2*, which are parallel versions of *gzip* and *bzip2*, respectively. For each utility we consider at least three compression levels: L – low, M – medium, and H – high.

To evaluate effectiveness of a networked file transfer, we need to determine the total time to complete the transfer. This time in general includes the following components: (i) sender overhead time; (ii) network connection setup time; (iii) file transmission time; and (iv) receiver overhead time. To measure effectiveness of data transfers, we use the effective throughput rather than the total transfer time. The effective upload or download throughput, measured in megabytes per second, is defined as the ratio between the uncompressed file size in megabytes and the time needed to complete the file transfer. This metric thus captures the system's ability to perform a file transfer in the shortest period of time regardless of a transfer mode.

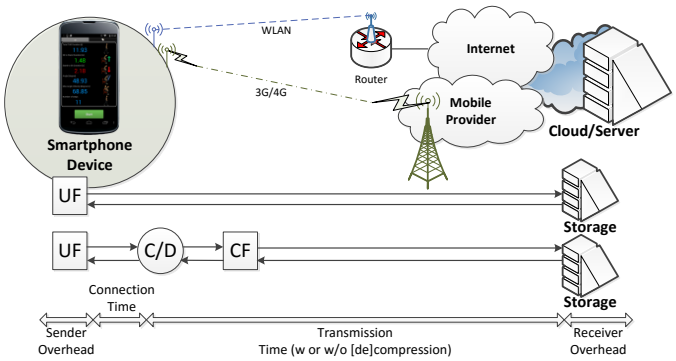

Fig. 1. Data transfers between mobile devices and the cloud

TABLE I
Compression Utilities

Utility	Compression levels	Version	Notes
gzip	1 – 9 (6)	1.6	DEFLATE (Ziv-Lempel, Huffman)
lzop	1 – 9 (3)	1.03	LZO (Lempel-Ziv-Oberhumer)
bzip2	1 – 9 (9)	1.0.6	RLE+BWT+MTF+RLE+Huffman
xz	0 – 9 (6)	5.1.0a	LZMA2
pigz	1 – 9 (6)	2.3	Parallel implementation of gzip
pbzip2	1 – 9 (9)	1.1.6	Parallel implementation of bzip2

The effective upload and download throughputs depend on many factors, including the file size and type, selected compression utility, the compression level, network characteristics such as latency and throughput, as well as the smartphone's performance. Whereas previous studies showed that compressed uploads and downloads can save time and energy in many typical file transfers initiated from smartphones [11], [14], [15], there is not a single transfer method that works the best for all data files and network conditions. To underscore this problem, we conduct a measurement-based study that evaluates effectiveness of various data transfer options under different network conditions.

B. Target Platforms

We use Google's Nexus 4 [16] and OnePlus One [17] smartphones as the target platforms during the case studies and the experimental evaluation of the proposed models. The Nexus 4 is powered by a Qualcomm Snapdragon S4 Pro (APQ8064) system-on-a-chip that features a quad-core ARM Cortex A15 processor running up to 1.512 GHz clock frequency and an Adreno 320 graphics processor and 2 GB of RAM memory. The OnePlus One is powered by a Qualcomm Snapdragon 801 (MSM8974AC) system-on-a-chip that features a quad-core ARM-based Krait 400 processor running up to 2.5GHz clock frequency, an Adreno 330 graphics processor and 3 GB of RAM memory. Both smartphones support a range of communication protocols including WLAN 802.11n, Bluetooth 4.0, and cellular networks.

The measurements are conducted using a setup described in [14]. The smartphone's operating system is upgraded to (a) add common compression utilities not readily supported, and

(b) to add utilities for managing performance measurements.

C. The Case for Optimizing File Transfers

In this section we show the results of a measurement based study that evaluates effectiveness of uncompressed and compressed file transfers initiated on a mobile device. An OnePlus One smartphone transfers data to and from a remote server over the Internet using its WLAN interface. To demonstrate the impact of network parameters, the measurements are performed when the WLAN throughput is set to 0.5 MB/s and 5 MB/s. The WLAN network throughput is controlled using the Linux *tc* (traffic control) utility. We show that a compression (utility, level) pair that achieves the maximum throughput changes as a function of network conditions and file size and type.

Upload Example. In this example we upload a text file with inertial sensor recordings captured on a wearable health monitor – Zephyr Technologies BioHarness 3 chest belt. This type of data is often uploaded on the cloud where more sophisticated processing can take place. For example, we can extract the subject's type and level of physical activity, detect posture transitions, or quantify upper body movements during standardized medical tests for mobility assessment. In this case the file captures an acceleration vector during subject's activities of daily living that include walking, driving, and office work. The accelerometer vector is sampled with the frequency of 100 Hz. The uncompressed file size is 30.88 MB. The experiment involves uncompressed and compressed file uploads. For each type of a transfer, the time to upload the file is measured to determine the effective upload throughput.

TABLE II shows the compression ratio (CR) and the effective upload throughputs for all types of file uploads. The two bottom rows show speedups in the effective throughput when comparing the best performing compressed upload to the uncompressed upload [*best/raw*] and to the compressed upload using *gzip* -6 [*best/gzip*-6], which is considered the default compression mode. All utilities achieve a relatively high compression ratio that ranges from 4.41 for *lzop* -1 to 18.92 for *xz* -9. The high compression ratio is due to redundancy in time stamps attached to each record. Typically, higher compression levels of a compression utility result in higher compression ratios, but unfortunately require more time to compress files, which results in lower effective throughputs. The uncompressed upload on a 0.5 MB/s network achieves the effective throughput of 0.53 MB/s. The compressed upload with *gzip* -6 achieves the effective throughput of 3.70 MB/s. The best effective throughput of 6.26 MB/s is achieved with *xz* -0. Thus, the compressed upload with *xz* -0 improves effective throughput 11.82 times over the uncompressed upload and 1.69 times over the compressed upload with *gzip -6*.

The uncompressed upload on a 5 MB/s network achieves the effective throughput of 4.52 MB/s and the compressed upload with *gzip* -6 achieves the effective throughput of only 2.36 MB/s. This means that the compressed upload with *gzip* -6 lowers the effective throughput relative to the uncompressed upload because the time needed to perform compression exceeds the time savings due to transferring smaller files. The best effective throughput of 18.31 MB/s is achieved with *gzip* -1. Thus, *gzip* -1 offers 4.05- and 7.75-fold improvements over the uncompressed upload and the compressed upload using *gzip* -6, respectively.

Download Example. In this example, we consider downloading an Android executable file for the Dropbox application (dropbox.tar). To prepare the input file, the original *apk* file, which is a zip derived container, is extracted into an uncompressed tar archive file. The uncompressed 69.31 MB file and all compressed versions of the file are made available on the server. The experiment involves uncompressed and compressed file downloads. For each transfer mode, the total time to get the uncompressed version of the file is measured to determine the effective throughput.

TABLE II
THROUGHPUT WHEN UPLOADING ACCEL.CSV

Utility & Level		CR	Throughput [MB/s]	
Net Thr. [MB/s]			**0.5 MB/s**	**5.0 MB/s**
gzip	1	6.22	3.18	18.31
gzip	6	**7.91**	**3.70**	**2.36**
gzip	9	8.57	0.48	0.57
lzop	1	4.41	2.31	15.24
lzop	6	4.41	2.29	15.46
bzip2	1	12.56	2.29	2.79
bzip2	6	11.97	2.00	2.08
bzip2	9	12.00	2.11	2.01
xz	0	12.93	6.26	9.81
xz	1	12.36	4.65	3.66
xz	6	18.91	0.25	0.26
xz	9	18.92	0.23	0.28
pigz	1	6.23	3.16	17.44
pigz	6	7.92	3.99	17.11
pigz	9	8.58	3.19	3.23
raw	-	1.00	0.53	4.52
[best/raw]	-	-	**11.82**	**4.05**
[best/gzip-6]	-	-	**1.69**	**7.75**

TABLE III
THROUGHPUT WHEN DOWNLOADING DROPBOX.TAR

Utility & Level		CR	Throughput [MB/s]	
Net Thr. [MB/s]		-	**0.5 MB/s**	**5.0 MB/s**
gzip	1	1.83	0.90	8.71
gzip	6	1.91	**0.95**	**9.00**
gzip	9	1.92	0.94	9.07
lzop	1	1.53	0.75	7.44
lzop	6	1.54	0.77	7.38
bzip2	1	1.92	0.96	7.12
bzip2	6	1.94	0.97	5.22
bzip2	9	1.94	0.97	4.54
xz	0	2.11	1.07	10.12
xz	1	2.16	1.07	10.26
xz	6	2.31	1.16	10.76
xz	9	2.58	**1.23**	**12.02**
pigz	1	1.84	0.87	8.58
pigz	6	1.92	0.96	9.21
pigz	9	1.93	0.96	9.19
raw	-	1.00	0.50	4.80
[best/raw]	-	-	**2.48**	**2.51**
[best/gzip-6]	-	-	**1.30**	**1.34**

TABLE III shows the compression ratio and the effective download throughputs for all types of data downloads. The two bottom rows show speedups in the effective throughput

when comparing the best performing compressed download with the uncompressed download and with the compressed download using *gzip* -6.

The uncompressed download on a 0.5 MB/s network achieves the effective throughput of 0.5 MB/s and the compressed download with *gzip* -6 achieves the effective throughput of 0.95 MB/s. The best effective throughput of 1.23 MB/s is achieved with *xz* -9. It offers 2.48- and 1.3-fold improvements over the uncompressed download and the compressed download using *gzip* -6, respectively.

The uncompressed download on a 5 MB/s network achieves the effective throughput of 4.8 MB/s and the compressed download with *gzip* -6 achieves the effective throughput of 9 MB/s. The best effective download throughput of 12.02 MB/s is achieved with *xz* -9. It offers 2.51- and 1.34-fold improvements over the uncompressed download and the compressed download using *gzip* -6, respectively.

These two examples demonstrate that not a single combination of a compression utility and a level offers the best throughputs in all conditions. The file size, file type, the level of data redundancy, device performance, and network conditions all impact the choice of best performing transfer mode. Moreover, these examples also show that the best performing transfer modes provide a substantial increase in the effective throughputs when compared to the uncompressed or the default compressed data transfers.

Ideally, we would like to design a framework for optimal file transfers between mobile devices and the cloud. The framework will autonomously, in real-time, with no significant overhead make a selection of a near optimal file transfer mode, while taking into account all parameters discussed above. To achieve this goal we need analytical models to support estimation of effectiveness of various transfer modes.

III. Modeling Uncompressed File Transfers

A. Models for Uncompressed File Transfers

The total time to perform a file transfer includes sender overhead time, network connection setup time, file transmission time, and receiver overhead time. In case of uncompressed file uploads, the sender and receiver overheads can be ignored. Thus, the total time of an uncompressed data file upload, *T.UUP*, includes the time to setup a network connection, *T.SC*, and the file transmission time, *T.UP*, as shown in Equation (1). If we know the network upload throughput, *Th.UP*, the file transmission time can be calculated by dividing the file size with the network upload throughput, *T.UP=US/Th.UP*. Similarly, the total time of an uncompressed data file download, *T.UDW*, includes *T.SC* and the file transmission time, *T.DW*, as shown in Equation (2). The file transmission time can be calculated as *T.DW=US/Th.DW*, where *Th.DW* is the network download throughput.

$$T.UUP = T.SC + T.UP = T.SC + US/(Th.UP) \quad (1)$$

$$T.UDW = T.SC + T.DW = T.SC + US/(Th.DW) \quad (2)$$

$$Th.UUP = \frac{Th.UP}{1 + Th.UP \cdot T.SC/US} \quad (3)$$

$$Th.UDW = \frac{Th.DW}{1 + Th.DW \cdot T.SC/US} \quad (4)$$

The effective upload throughput is calculated as the uncompressed file size divided by the total time to upload the file, *Th.UUP=US/T.UUP*. The effective download throughput is calculated as the uncompressed file size divided by the time to download the file, *Th.UDW=US/T.UDW*. Equations (3) and (4) show the expressions for the effective upload and download throughputs, respectively, as the functions of the file size, the time to set up the network connection, and the network upload and download throughputs. The effective throughputs, *Th.UUP [Th.UDW]*, reach the network throughputs, *Th.UP [Th.DW]*, when transferring very large files. In case of smaller files, the time to setup the network connection limits the effective throughput.

B. Model Verification

To verify the models for uncompressed file transfers we perform a set of measurement-based experiments as follows. An OnePlus One smartphone is used to initiate a series of file uploads to and downloads from a remote server. The smartphone is connected to the Internet over its WLAN interface. File transfers take place over a secure shell (*ssh*) - an encrypted network protocol. The file sizes are set to vary from 1 kB to 100 MB. The total transfer time is measured for each file transfer and the effective throughput is calculated. The upload and download experiments are repeated for four distinct network throughputs, set to 0.5, 2.0, 3.5, and 5.0 MB/s.

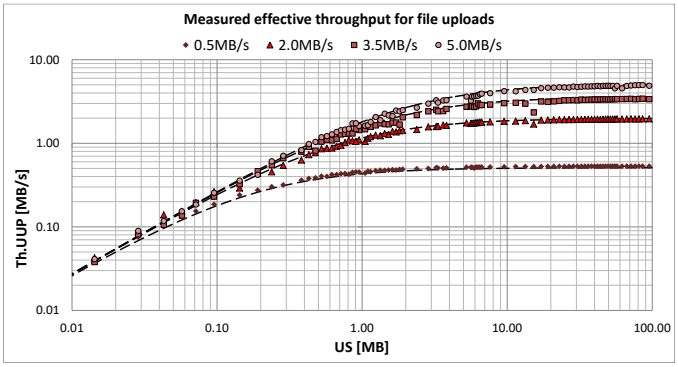

Fig. 2. Measured effective throughput for file uploads

Fig. 2 shows the effective throughput for uncompressed uploads as a function of the file size and the network connection throughput. The plots show that the effective throughput saturates for the larger files, reaching the network connection throughput, i.e., *Th.UUP=Th.UP*. By using a curve fitting software [18], we derive an equation that models the effective throughput. The dashed lines in Fig. 2 illustrate the derived equations for different network upload throughputs. The derived equations match the Equation (3) from the proposed analytical model with two constants corresponding to *T.SC* and *Th.UP*. The curve fitting software derives a constant that

corresponds to the time to setup the connection. For the setup used in our experiment *T.SC* is 0.39 seconds.

Fig. 3 shows the measured effective throughput for uncompressed file downloads for different network throughputs as a function of the file size. The results of the download experiments confirm the correctness of the proposed analytical models for the effective throughput. The derived constant for *T.SC* matches the one from the upload experiments.

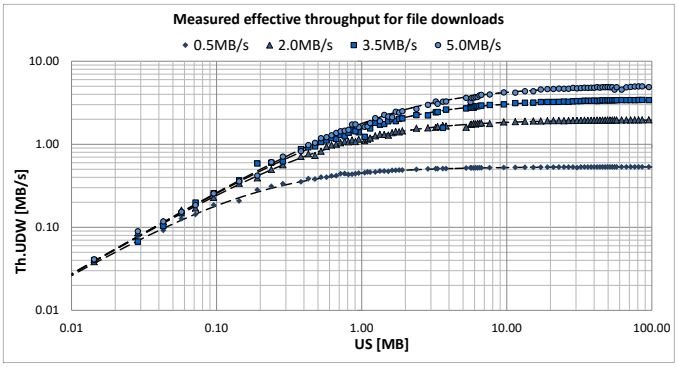

Fig. 3. Measured effective throughput for file downloads

C. Network Connection Characterization

The experimental verification of the models for the effective throughput requires a series of uploads and downloads of data files of different sizes. However, such an approach is time and resource consuming and thus not practical. Here we describe a practical method for deriving unknown network parameters (*Th.UP*, *Th.DW*, and *T.SC*) using the verified analytical model and a limited number of file transfers.

The proposed method involves performing a two file upload or download test. Two files of different sizes are selected to be transferred over a network connection with unknown parameters. The total transfer time is measured and used to calculate the effective throughput. These measured quantities are then used with the models to derive the unknown parameters.

To demonstrate deriving the network parameters, we consider file uploads over an *ssh* network connection that utilizes the smartphone's WLAN interface. We select two test files with sizes *US(s)*=0.14 MB and *US(l)*=1.24 MB. The measured effective upload throughputs are *Th.UUP(s)*=0.36 MB/s and *Th.UUP(l)*=2.06 MB/s. Next, by replacing the file sizes and the measured effective throughputs in Equation (5) we get two equations with two unknowns, *T.SC* and *Th.UP*. By solving the system of linear equations, shown in Equation (6), we derive *Th.UP*=5.167 MB/s and *T.SC*=0.362 seconds.

Fig. 4 illustrates the proposed method. The measured upload throughputs for two selected files are marked with a blue and a red diamond. By deriving *Th.UP* and *T.SC* as described above, the model from Equation (3) is plotted using a black dashed dot curve. The actual measurements of the effective upload throughputs performed during the verification phase are shown as blue circles. A visual inspection shows that the model with parameters extracted by just two measurements matches the actual measurements performed during the verification phase.

$$Th.UP = \frac{Th.UUP}{1 - Th.UUP \cdot T.SC/US} \tag{5}$$

$$\begin{aligned} Th.UP - 2.57 \cdot Th.UP \cdot T.SC &= 0.36 \\ Th.UP - 1.66 \cdot Th.UP \cdot T.SC &= 2.06 \end{aligned} \tag{6}$$

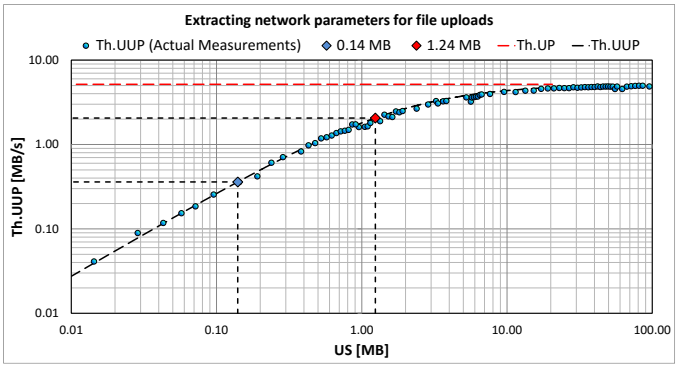

Fig. 4. Extracting network parameters for file uploads

IV. Modeling Compressed File Transfers

A. Compressed Uploads

A compressed upload of a data file from a mobile device to the cloud can be performed in two ways, sequentially or with the use of piping. In the former, the data file is first compressed locally on the mobile device and then the compressed file is transferred to the cloud, with no overlap between these two tasks. In the later, the file compression is partially or completely overlapped by setting up the network connection and the file transmission. Thus, we can determine the upper and lower limits for the total compressed upload time. The maximum compressed upload time shown in Equation (7), *T.CUP.max*, includes the time to perform the local compression of the file on the mobile device, *T.C*, the time to setup network connection, *T.SC*, and the time to transfer the compressed file, *T.CUP'*. The minimum upload time shown in Equation (8), *T.CUP.min*, includes the time to setup network connection and the time to transfer the compressed file of size. The time to transfer the compressed file can be calculated as the compressed file size, which is *US/CR*, where *CR* is the compression ratio, divided by the network connection upload throughput *Th.UP*. The local compression throughput, *Th.C*, is defined as the uncompressed file size divided by the time to perform a local compression *Th.C=US/T.C*. This "higher is better" metric captures ability of a mobile device to perform local compression fast.

The minimum upload throughput, *Th.CUP.min*, is calculated as the uncompressed file size in megabytes, *US*, divided by the maximum time to perform compressed upload as shown in Equation (9). The maximum upload throughput, *Th.CUP.max*, is calculated as the uncompressed file size in megabytes, *US*, divided by the minimum time to perform compressed upload as shown in Equation (10). The final expressions in Equations (11) and (12) show the boundaries for the compressed upload throughputs as a function of the network parameters, *Th.UP*, *T.SC*, file size, *US*, compression ratio, *CR*, and the local com-

pression throughput, *Th.C*. From these expressions, we can analytically estimate the impact of changes in these parameters. For example, the highest compressed upload throughput that can be achieved approaches the product of the compression ratio and the network connection upload throughput, which is possible in devices where local compression throughputs exceed the network upload throughput and when the size of a transferred file is sufficient to minimize the effects of the network connection setup time.

$$T.CUP.max = T.C + T.SC + T.CUP' \quad (7)$$

$$T.CUP.min = T.SC + T.CUP' \quad (8)$$

$$Th.CUP.min = \frac{US}{T.CUP.\max} \quad (9)$$

$$Th.CUP.max = \frac{US}{T.CUP.\min} \quad (10)$$

$$Th.CUP.min = \frac{CR \cdot Th.UP}{1 + Th.UP \cdot CR \cdot (\frac{1}{Th.C} + \frac{T.SC}{US})} \quad (11)$$

$$Th.CUP.max = \frac{CR \cdot Th.UP}{1 + Th.UP \cdot CR \cdot T.SC/US} \quad (12)$$

Fig. 5 illustrates the estimated minimum and maximum throughputs, *Th.CUP.min* and *Th.CUP.max*, respectively, as well as the measured compressed upload throughput, *Th.CUP*, for different modes of compressed upload. The measurements are performed on Nexus 4 smartphone with a 2.5 MB/s WLAN network interface. The measured compressed upload throughput is between the predicted minimum and maximum throughputs. For example, the estimated lower and upper limit for the compression throughput of *gzip* with -1 are 3.9 MB/s and 6.2 MB/s, and the measured compression throughput is 5.9 MB/s; in contrast, the estimated bounds for *bzip2* with -1 are 1.8 MB/s and 8.1 MB/s and the measured compression throughput is 2.04 MB/s. In cases when the local compression throughput falls below the network connection upload throughput, *Th.C* << *Th.UP*, the effective compressed upload throughput is closer to the minimum throughput (e.g., for *xz*). In cases when *Th.C* >> *Th.UP*, the effective compressed upload throughput is closer to the expected maximum throughput (e.g, for *lzop*).

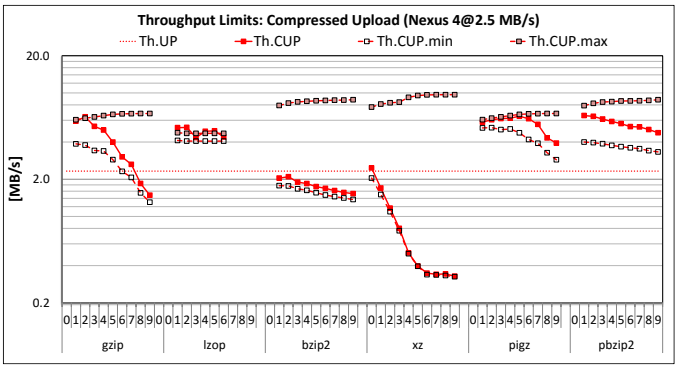

Fig. 5. Effective compressed upload throughputs

B. Compressed Downloads

A compressed download from the cloud, initiated from a mobile device, can be done sequentially or with the use of piping. In the former, the compressed data file is downloaded on the mobile device and then the compressed file is decompressed with no overlap between these two tasks. In the later, the file decompression is partially or completely overlapped by the compressed file transmission. Thus, we can determine the limits for the total download time. The maximum total download time shown in Equation (13), *T.CDW.max*, includes the time to setup network connection, *T.SC*, the time to transfer the compressed file, *T.CDW'*, and the time to perform the decompression of the received file on the mobile device, *T.D*. The minimum download time shown in Equation (14), *T.CDW.min*, includes the time to setup network connection and the time to transfer the compressed file. The time to transfer the compressed file can be calculated as the compressed file size, *US/CR*, divided by the network connection download throughput *Th.DW*. The time to perform decompression on the mobile device is used to determine the local decompression throughput, *Th.D*, which is defined as the uncompressed file size divided by the time to perform decompression, *US/T.D*. This metric thus captures the mobile device's ability to effectively perform decompression.

$$T.CDW.max = T.D + T.SC + T.CDW' \quad (13)$$

$$T.CDW.min = T.SC + T.CDW' \quad (14)$$

$$Th.CDW.min = \frac{US}{T.CDW.\max} \quad (15)$$

$$Th.CDW.max = \frac{US}{T.CDW.\min} \quad (16)$$

$$Th.CDW.min = \frac{CR \cdot Th.DW}{1 + Th.DW \cdot CR \cdot (\frac{1}{Th.D} + \frac{T.SC}{US})} \quad (17)$$

$$Th.CDW.max = \frac{CR \cdot Th.DW}{1 + Th.DW \cdot CR \cdot T.SC/US} \quad (18)$$

The minimum effective compressed download throughput, *Th.CDW.min*, is calculated as the uncompressed file size in megabytes divided by the maximum time to perform compressed upload, as shown in Equation (15). The maximum download throughput, *Th.CDW.max*, is calculated as the uncompressed file size divided by the minimum time to perform the compressed download as shown in Equation (16). The final expressions in Equations (17) and (18) show the boundaries for the compressed download throughputs as a function of the network parameters, file size, compression ratio, and the local decompression throughput.

Fig. 6 illustrates the estimated throughput boundaries and the measured compressed download throughput for different modes of compressed download. The measurements are performed on Nexus 4 smartphone with a 2.5 MB/s WLAN network interface. The measured compressed download throughput is between the predicted minimum and maximum

throughputs. For example, the estimated lower and upper boundaries for the decompression throughput of *gzip* with -9 are 6.19 MB/s and 7.29 MB/s, and the measured compression throughput is 7.16 MB/s. The utilities with high local decompression throughputs achieve the effective download throughputs close to the upper boundaries when downloading large files (e.g., *gzip* and *lzop* for all compression levels).

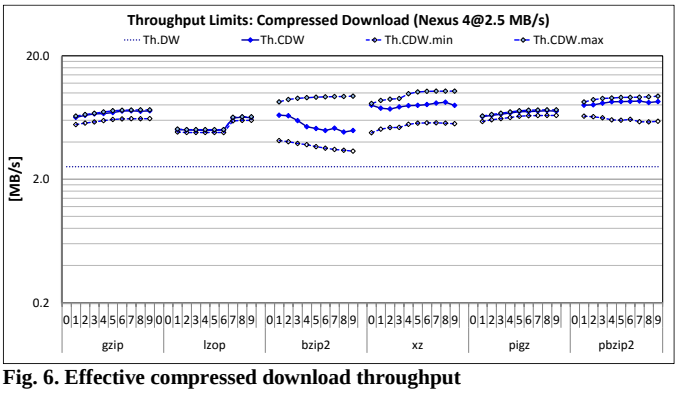

Fig. 6. Effective compressed download throughput

C. *Piping Model*

Whereas we experimentally verified that we can estimate the minimum and maximum compressed upload and download throughputs, the distance between these boundaries for a particular compression utility is often too wide, rendering this model insufficient to estimate the effective throughputs. We would like to be able to devise models for accurate estimation of effective upload and download throughputs.

The use of piping when transferring a file is beneficial as it increases the effective throughput. It allows for overlapping local (de)compression tasks with the file transfer task on mobile devices. In case of compressed upload, a degree of this overlap depends on the ratio between the network upload throughput and the local compression throughput. When the local compression throughput exceeds by far the network upload throughput, the bottleneck is the network. When the local compression throughput falls below the network throughput, the compressed uploads are not beneficial. To derive the piping model, the local compression term from the lower throughput limit is restricted using a ratio, *k.th.c*, described in Equation (19). This factor lowers the impact of the local compression term when the local compression throughput exceeds the network connection upload throughput. The final piping model for the compressed upload throughput is shown in Equation (20).

Fig. 7 shows the estimated compressed upload throughput (green dots) and the measured compressed upload throughput (red squares) for all considered compression utilities and compression levels when uploading an input file from Nexus 4 to the server over a 2.5 MB/s WLAN connection. The plot suggests a very high accuracy of the proposed model for all compression utilities and compression levels. This expression implies that if we know the parameters of the network connection (*Th.UP* and *T.SC*), and if for a given file of size *US* we can predict the compression ratio, *CR*, and local compression throughput for a given (utility, level) (*Th.C*), we can fairly accurately estimate the expected compressed upload throughput using the piping model.

$$k.th.c = \begin{cases} Th.UP/Th.C, & Th.C > Th.UP \\ 1, & Th.C < Th.UP \end{cases} \quad (19)$$

$$Th.CUP.pipe \approx \frac{CR \cdot Th.UP}{1 + CR \cdot Th.UP \cdot (\frac{T.SC}{US} + \frac{k.th.c}{Th.C})} \quad (20)$$

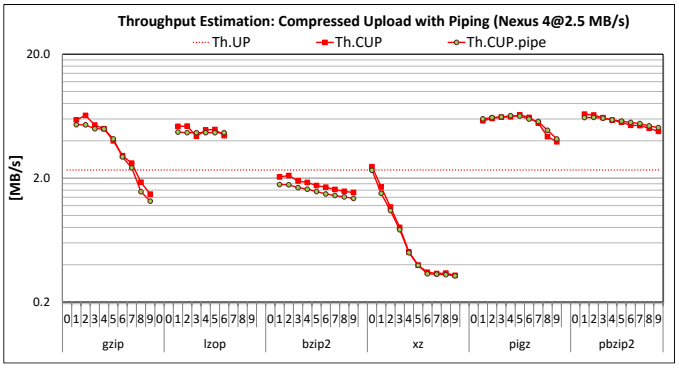

Fig. 7. Throughput estimation – compressed upload with piping

When piping is used for compressed data download, the degree of overlapping depends on the ratio between the network download throughput, *Th.DW*, and the local decompression throughput, *Th.D*. To derive the piping model, the decompression term from the lower throughput limit is restricted using a corrective factor, *k.th.d*, described in Equation (21). This factor lowers the impact of the local decompression term when the local decompression throughput exceeds the network download throughput. The final piping model for compressed download throughput is shown in Equation (22).

$$k.th.d = \begin{cases} Th.DW/Th.D, & Th.D > Th.DW \\ 1, & Th.D < Th.DW \end{cases} \quad (21)$$

$$Th.CDW.pipe \approx \frac{CR \cdot Th.DW}{1 + CR \cdot Th.DW \cdot (\frac{T.SC}{US} + \frac{k.th.d}{Th.D})} \quad (22)$$

Fig. 8 shows the estimated compressed download throughput (green circles) and the measured compressed download throughput (blue triangles) for all considered compression utilities and compression levels when downloading an input file from the server to the Nexus 4 over the 2.5 MB/s WLAN connection. The plot suggests a very high accuracy of the proposed model for all compression utilities and compression levels. This expression implies that if we know the parameters of the network connection (*Th.DW* and *T.SC*), and if for a given file of size *US* we can predict the compression ratio, *CR*, and local decompression throughput for a given utility, level pair (*Th.D*), we can fairly accurately estimate the expected compressed download throughput in the piping model.

The proposed models rely on three sets of parameters: those that are readily available (e.g., file size), those that can be determined using simple experiments (*T.SC*, *Th.UP*, *Th.DW*),

and those that are unknown such as the compression ratio, *CR*, and compression or decompression throughput, *Th.C [Th.D]*. To be able to successfully apply and use the proposed models, the compression ratio and the time it takes to perform compression or decompression of files have to be estimated. One method which can provide estimation for compression ratio and local (de)compression throughputs is the use of data tables filled with historical data of prior data transfers and their effectiveness for specific compression utilities and levels.

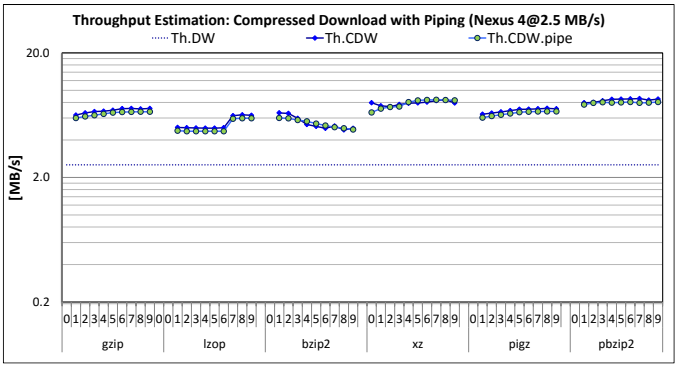

Fig. 8. Throughput estimation – compressed download with piping

D. *Model Verification and Estimation Errors*

To verify the piping models and to quantify their ability to estimate effective throughputs, we perform a set of measurement-based experiments. For each upload or download file transfer, the effective throughput is estimated using the piping model. In addition, the total transfer time is measured and then used to determine the measured effective throughput. An estimation error is calculated as the difference between the measured and the estimated throughput, divided by the measured throughput. Two data sets representative for uploads and downloads initiated on mobile platforms are used in the experiments. The file transfers are initiated on the OnePlus One smartphone with a 0.5 MB/s and 5 MB/s WLAN network throughputs.

Upload. For upload, we use a data set that includes over 40 files containing physiological data recorded by a wearable health monitor - Zephyr Technologies BioHarness 3 chest belt. The files are in text format and range from 0.5 MB to 300 MB in size.

Fig. 9 shows the estimation errors for compressed uploads on the low network throughput of 0.5 MB/s for all considered compression utilities with compression level -1. For transfers of large files (10 MB to 300 MB), the estimation errors are less than 1% for all utilities except *xz* and *bzip2*, which result in slightly higher estimation errors of up to 5% and -40%, respectively. For transfers of small files, the estimation errors start at 10% at 0.1 MB file size and fall down to less than 1%.

Fig. 10 shows the estimation errors for compressed uploads on the high network throughput of 5 MB/s for all considered compression utilities with compression level -1. For transfers of large files (10 MB to 300 MB), the estimation errors are from 10% to 5% for all utilities except *xz*, which results in estimation errors of up to -40%. For smaller files, the estimation error starts at 50% and falls down to 10%. For both network throughputs, the error boundaries are higher for compressed uploads of small files due to the smaller execution time, which makes estimation more sensitive to the measurement variability in the verification experiments.

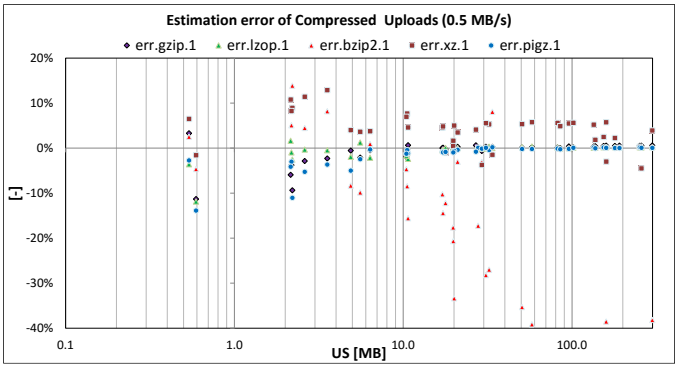

Fig. 9. Estimation error of compressed uploads (0.5 MB/s)

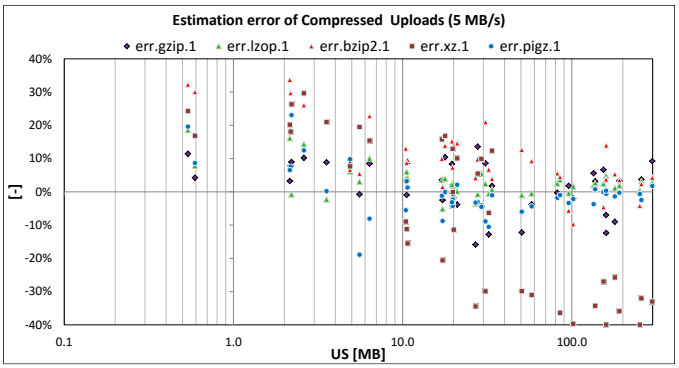

Fig. 10. Estimation error of compressed uploads (5 MB/s)

Download. For download, we use a dataset that includes over 140 files containing executables of popular Android applications. To prepare the input files, the original *apk* files are extracted into an uncompressed tar archive files. The files range from 0.1 MB to 150 MB in size.

Fig. 11 shows the estimation errors for compressed downloads on the low network throughput of 0.5 MB/s for compression utilities with compression level -9. When downloading large files (10 MB to 150 MB), the estimation errors are between 3% and 1% for all compression utilities. When transferring small files the estimation error starts at 16% at 0.1 MB file size and falls down to 3%.

Fig. 12 shows the estimation errors for compressed downloads on the high network throughput of 5 MB/s for all compression utilities with compression level -9. When downloading large files, the estimation error is around 22-15% for all compression utilities. When transferring small files, the estimation errors start at 33% and fall down to 20-15%. For both network throughputs, the error boundaries are higher for compressed downloads of small files due to their lower execution times, which make estimation more sensitive to the measurement variability.

Overall, the analytical models provide an accurate prediction of effective throughput of compressed data transfers over varying files sizes and with a variability of the network pa-

rameters. The estimation error is increasing for higher throughput and smaller file sizes, which can be attributed to the higher sensitivity of conducted experiments to the measurement variability of local, uncompressed and compressed throughputs.

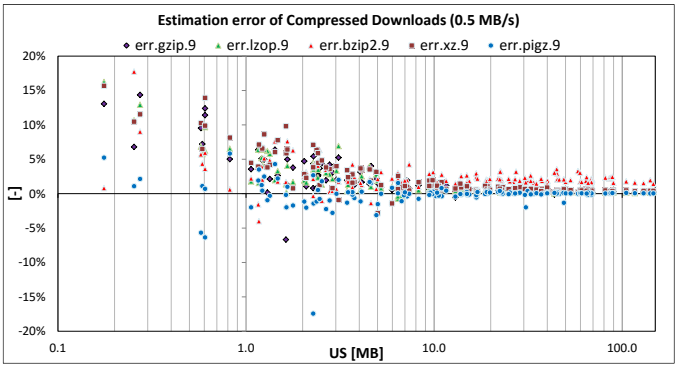

Fig. 11. Estimation errors for compressed downloads (0.5 MB/s)

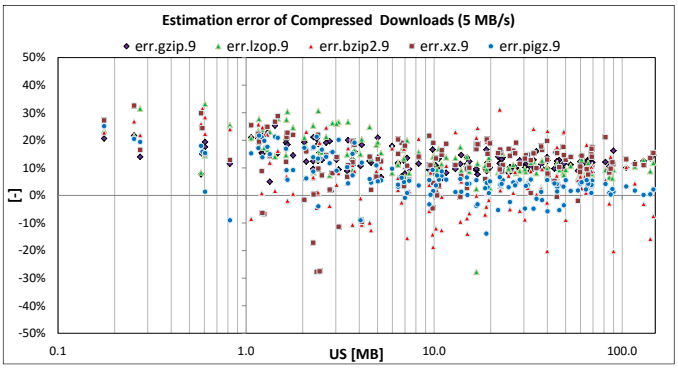

Fig. 12. Estimation errors for compressed downloads (5 MB/s)

V. Conclusions

This paper introduces analytical models for characterizing effective throughput of uncompressed and compressed data file transfers between mobile devices and the cloud. We have demonstrated the validity of the models through the series of measurement-based experiments conducted on Nexus 4 and OnePlus One smartphones. The experiments have demonstrated that the proposed models can accurately estimate throughputs if we know the parameters of network connection, and if for a given file we can predict the compression ratio and local (de)compression throughputs.

Using the proposed analytical models, we can initiate the development of frameworks for optimizing data transfers between mobile devices and the cloud. The framework can be designed to be conscientious of the network conditions, the user's history of data transfers (type and size of files transfers, a frequency of transfers), the file characteristics, available compression utilities, and their performance profiles.

The future research will focus on the development of models for energy efficiency in compressed transfers and on the estimation of energy efficiency from the described throughput estimations of uncompressed and compressed transfers.

References

[1] CISCO, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper." 03-Feb-2015.

[2] V. Agababov, M. Buettner, V. Chudnovsky, M. Cogan, B. Greenstein, S. McDaniel, M. Piatek, C. Scott, M. Welsh, and B. Yin, "Flywheel: Google's Data Compression Proxy for the Mobile Web," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2015, pp. 367–380.

[3] Google, "Data Server - Google Chrome," 2014. [Online]. Available: https://developer.chrome.com/multidevice/data-compression. [Accessed: 30-Oct-2015].

[4] Amazon, "What Is Amazon Silk? - Amazon Silk," 2015. [Online]. Available: http://docs.aws.amazon.com/silk/latest/developerguide/. [Accessed: 06-Dec-2015].

[5] Onavo, "Onavo," *Onavo*, 2015. [Online]. Available: http://www.onavo.com. [Accessed: 01-Dec-2015].

[6] Snappli, "Snappli," 2014. [Online]. Available: http://snappli.com/. [Accessed: 01-Dec-2015].

[7] zlib, "zlib Home Site," 2015. [Online]. Available: http://www.zlib.net/. [Accessed: 16-Dec-2015].

[8] Google, "About Attachment Manager," 2014. [Online]. Available: http://www.google.com/support/enterprise/static/postini/docs/admin/en/admin_msd/attach_overview.html. [Accessed: 31-Oct-2015].

[9] A. Dzhagaryan, A. Milenkovic, and M. Burtscher, "Energy efficiency of lossless data compression on a mobile device: An experimental evaluation," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, Austin, TX, 2013, pp. 126–127.

[10] A. Milenkovic, A. Dzhagaryan, and M. Burtscher, "Performance and Energy Consumption of Lossless Compression/Decompression Utilities on Mobile Computing Platforms," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, 2013, pp. 254–263.

[11] A. Dzhagaryan and A. Milenkovic, "On Effectiveness of Lossless Compression in Transferring mHealth Data Files," in *2015 IEEE 17th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Boston, MA, 2015.

[12] K. Barr and K. Asanović, "Energy aware lossless data compression," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys '03)*, 2003, pp. 231–244.

[13] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Transactions on Computer Systems*, vol. 24, no. 3, pp. 250–291, Aug. 2006.

[14] A. Dzhagaryan, A. Milenković, and M. Burtscher, "Quantifying Benefits of Lossless Compression Utilities on Modern Smartphones," in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, Las Vegas, NV, 2015.

[15] A. Milenkovic, A. Dzhagaryan, and M. Burtscher, "Performance and Energy Consumption of Lossless Compression/Decompression Utilities on Mobile Computing Platforms," in *2013 IEEE 21st International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, San Francisco, CA, 2013, pp. 254–263.

[16] Google, "Nexus - Google," 2014. [Online]. Available: http://www.google.com/intl/all/nexus. [Accessed: 15-Jun-2014].

[17] OnePlus, "OnePlus One," 2015. [Online]. Available: https://oneplus.net/one. [Accessed: 12-Jul-2015].

[18] "CurveExpert and GraphExpert Software," 2015. [Online]. Available: https://www.curveexpert.net/. [Accessed: 07-May-2016].