# Improving the QoE in Personal Clouds with Cross-Swarm Bundling

Rahma Chaabouni*, Marc Sánchez-Artigas*, Ala Chaabouni‡* and Pedro García-López*
*Universitat Rovira i Virgili, Tarragona (Spain)
{rahma.chaabouni|marc.sanchez|pedro.garcia}@urv.cat
‡National School of Engineers of Sfax, University of Sfax, Sfax (Tunisia)
alaa.chaabouni@enis.tn

***Abstract*—Personal cloud storage systems, like Dropbox, are revolutionizing the way people think about and access their files. As the prevailing model, these systems use unicast to push file changes to each of the "unsynced" devices. And as a result, they transmit multiple times the same information, once per unsynced device. This puts an unnecessary strain on outgoing bandwidth at the datacenters. One way to address this is to leverage P2P-like content distribution to benefit from user resources at the edges of the Internet.**

**Although protocols like BitTorrent have proven to be effective in this scenario, we go a step further in this work and propose *cross-swarm bundling* as a mechanism for file distribution. One key contribution of this work is that, instead of using bundling as means to extend the lifetime of swarms, we show that it can be useful to improve the *Quality of Experience* (QoE). We validate our proposal using a trace of Ubuntu One, a real personal cloud system, obtaining significant improvements on the QoE levels.**

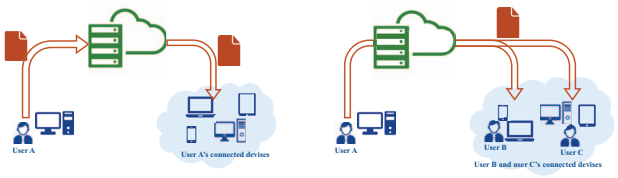
***Index Terms*—Personal Clouds, QoE, Cross-swarm bundling, BitTorrent**

## I. Introduction

Cloud storage services such as Dropbox, Box and Google Drive have become increasingly popular. These services, also referred to as *Personal Clouds*, offer three key services to users: *Storage*, *Synchronization* and *Sharing*, the 3S definition [10]. Because of their prominent role in the Internet, there has been a surge of research on the benchmark and design of these systems [6], [5], [11], [18], [19].

For file synchronization and sharing (see Fig.1), the *de facto* standard to distribute files, "deltas" or "diffs", to each of the out-of-sync devices is HTTP unicast. From a network perspective, this means that the same content is sent multiple times, once per device, putting an unnecessary strain on the instantaneous outgoing bandwidth at datacenters. In practice, this may translate into a loss of Quality of Service (QoS) when the number of parallel transmissions is high, in particular, at peak hours.

One way to address this issue is to leverage P2P-like content distribution to benefit from user resources at the edges of the Internet. In this sense, our previous works [2] and [3] have proven BitTorrent [4] to be effective in decreasing outbound traffic, despite the small size of data flows [12] and swarms [2]. By benefiting from the mutual need of devices to synchronize, [2] and [3] *exploited their aggregate upload bandwidth to offload the cloud from doing all the serving.*




(a) Synchronization (b) Sharing

Fig. 1: Synchronization and sharing in personal cloud systems

In this research, we focus on a new dimension, the Quality of Experience (QoE), and set as our main goal the reduction of the delay experienced by devices to get the necessary files and deltas to be "in sync" again. To achieve this goal, BitTorrent per se may be not sufficient, as the total number of unsynced devices per file is generally small [2], which poses a limit on the maximum attainable benefits.

A good solution to this problem is to "inflate" the swarms by grouping a set of diverse contents into a single `.torrent` file. This is known as *bundling* in BitTorrent parlance, and was originally proposed to mitigate the problem of availability in unpopular torrents [20], [14]. In personal clouds, low content availability is not an issue, as the cloud is always available to serve the files.

However, *cross-swarm bundling* can have a positive effect on the QoE in personal clouds, and clouds in general. As far as we know, we are the first to provide evidence that bundling can be useful to diminish the download time, even in a very adverse scenario such as personal clouds, where most of files are small. More specific to our problem is the investigation of the benefits of inflating swarms with HTTP users or even of the merge of two HTTP users to create a BitTorrent swarm. We also study the efficiency of cross-swarm bundling in personal clouds through an average case analysis and identify the cases where it is positive. Furthermore, we show how to implement bundling in personal clouds. Finally, we validate our proposal using a real trace of the *Ubuntu One* system [12] and compare the different approaches that can be deployed to deliver content from the cloud to end users. Our results demonstrate that when the load is high, bundling can improve significantly the QoE of users without increasing the datacenter bandwidth.

The remainder of the paper is organized as follows: First we


DOI 10.1109/LCN.2016.71

discuss related work in Section II. In Section III, we describe briefly the architecture of personal clouds and state the QoE problem. In Sections IV and V, we present our cross-swarm bundling model along with an average case analysis on the efficiency of bundling in personal clouds. Section VI presents the implementation details of bundling in personal clouds. We validate our proposal, present and discuss the results in Section VII. Finally, Section VIII concludes the paper and presents our future plans.

## II. Related work

The idea of using BitTorrent in personal clouds was introduced with the objective of reducing the load on the cloud [1]. It was proposed to transparently switch from HTTP to BitTorrent upon detection of a certain critical mass demand on a specific content. A threshold was statically placed on the number of users requesting the same file, and the system had to test with each new request whether the current number of requesters of the corresponding file passes the predefined threshold or not. The download time and offload ratio in BitTorrent compared to HTTP were studied in [2], and a dynamic algorithm for the management of these protocols was proposed [3]. The choice of the protocol was made based on the prediction of the efficiency of BitTorrent and HTTP for each case. The main goal of [1], [2], [3] was to reduce the bandwidth consumption in the cloud. Nevertheless, the number of simultaneous downloads of the same content remains very limited in personal clouds. This is due to the fact that a large population of the clients use these services for backup. In this paper, we leverage the clients resources at maximum by enabling cooperation between peers of different swarms through cross-swarm bundling.

Bundling in BitTorrent systems has gained the attention of researchers as it has been reported to increase the availability of unpopular files. In this context, the authors in [20] performed a measurement study on real BitTorrent swarms which showed that availability is a serious issue. They proposed a model to quantify content availability in swarming systems and used it to analyze the implications of bundling. Their results proved the efficiency of this technique in improving the availability of unpopular torrents and reducing the download times when publishers are unavailable. However, no related work focused on the effect of bundling on the QoE when availability is not an issue (which is the case in personal clouds).

Zhan et al. presented in [22], [21] a good motivation section on the importance of bundling. They propose the design, proof-of-concept and implementation of a dynamic file bundling system. Han et al. performed a large scale measurement study on the prevalence of file bundling in the BitTorrent ecosystem [15], [16]. They noticed that over $70\%$ of the monitored torrents contain multiple files which proves that bundling is very common. However, this type of measurement is only related to static bundling when all the files are grouped initially by the publisher.

The concept of bundling in personal clouds is different from the one in swarming systems. As a matter of fact, bundling in personal clouds consists in grouping the files that need to be transferred to the same user into a single one, such that both transmission latency and control overhead are reduced. Drago et al. [6], [5] compared the performance of 5 popular personal clouds and noticed that, among the personal clouds in their study, only Dropbox implements a bundling strategy. However, their measurement study lacks more details about the concrete implementation details and the bundling policies in Dropbox.

## III. Problem statement: QoE in personal clouds

*Personal Cloud* is a term generally used to refer to a file hosting service that allows its users to store, synchronize and share content over the Internet. When a user adds a new file to his personal space in the cloud, all his out-of-sync devices will be notified to download this new content in a unicast form. During peak hours when the number of download requests is high, this approach can lead to overloading the cloud, thus increasing the download times for the clients. To overcome these issues, the cloud can benefit from the clients' upload capacities. This can be done by using BitTorrent to distribute the files that are shared between a set of devices.

The use of BitTorrent in personal clouds implies grouping devices into swarms. We define a *swarm* by the set of devices (called also clients or peers) that are requesting the same file. We differentiate between two types of swarms: *HTTP swarms* and *BitTorrent swarms*, based on the download protocol used. If a file is being downloaded by a single peer, we consider it as a single-peer swarm.

In this paper, we aim at *improving the QoE for the clients without increasing the cloud's bandwidth consumption*. The QoE, also known as the perceived quality of service, measures the user's satisfaction with the provided service. Several parameters can interfere in evaluating the QoE. In this paper, we focus on the download time observed by the clients. For each request, we compare the download times measured using HTTP only and using bundling. The approach that entails less download time is the one that offers a better QoE.

We denote by $\Delta QoE$ the difference (in seconds) between the download time experienced by a given user downloading a file$f$ with HTTP only and the download time experienced by the same user downloading the same file after being bundled with another swarm. A positive value of $\Delta QoE$ reflects an improvement in download time compared to the use of HTTP only, while a negative $\Delta QoE$ means that the use of bundling has affected negatively the download time and made it longer. $\Delta QoE = 0$ when there is no change in the download times.

**Goal**. The aim of this paper is to maximize $\Delta QoE$ for users, which requires an answer to the questions of *how to decide which swarms should be grouped together among the large set of swarms, and which are the criteria for making such a decision?* .

TABLE I: Table of notation

| Symbol | Meaning |
|---|---|
| $s$ | a swarm: a collection of peers downloading the same file $f_s$. |
| $f_s$ | the file being downloaded by the peers in $s$. |
| $F_s$ | size of the file $f_s$. |
| $w_s$ | amount of cloud bandwidth allocated to $s$. |
| $L_s$ | number of peers in $s$. |
| $d_{min,s}$ | the download speed of the slowest peer in $s$. |
| $u_s$ | the average upload speed of the peers in $s$. |
| $\eta_s$ | the effectiveness of file sharing[1]. $\eta_s \in [0,1]$ where 1 means maximum effectiveness while 0 reflects the absence of collaboration between peers. |
| $\alpha_{bt}$ | the overhead related to the start-up phase in BitTorrent transfers[1]. |
| $t_s^{http}$ | the expected end download time for the peers in $s$ when HTTP is the download protocol. |
| $t_s^{bt}$ | the expected end download time for the peers in $s$ when BitTorrent is the download protocol. |

## IV. Cross-swarm bundling model

In this section, we describe our cross-bundling model. We also present some mathematical background to estimate the expected end download time for a given swarm.

### A. Background: The expected end download time

In this section, we reuse the equations proposed in [2], which predict the expected end download time for a given swarm $s$ based on the adopted download protocol. The complete notation list is available in Table I.

$$t_s^{http} = \frac{F_s}{\min\left\{d_{min,s}, \frac{w_s}{L_s}\right\}}. \quad (1)$$

$$t_s^{bt} = \frac{F_s}{\min\left\{d_{min,s}, \frac{w_s + \eta_s\, u_s\, L_s}{L_s}, w_s\right\}} + \alpha_{bt}. \quad (2)$$

(1) measures $t_s^{http}$, the expected end download time for a swarm $s$ using HTTP as a download protocol. $t_s^{http}$ is limited by the bandwidth bottleneck of the transfer. This bottleneck can be either the download speed of the slowest peer in the swarm or the share of cloud upload bandwidth allocated to each peer. (2) measures $t_s^{bt}$, the expected end download time if the peers in $s$ are using BitTorrent as a download protocol. This equation[1] comes from a fluid model proposed first by Kumar et al. in [17] and extended later in [2]. $t_s^{bt}$ is limited by the download speed of the slowest peer, or the rate at which "fresh" data can be sent to the different peers. This rate can be equal to the upload speed of the seed or the aggregate upload of the system divided between all the peers.

[1]For further information about the formulas and the related parameters, we kindly refer the reader to [2].

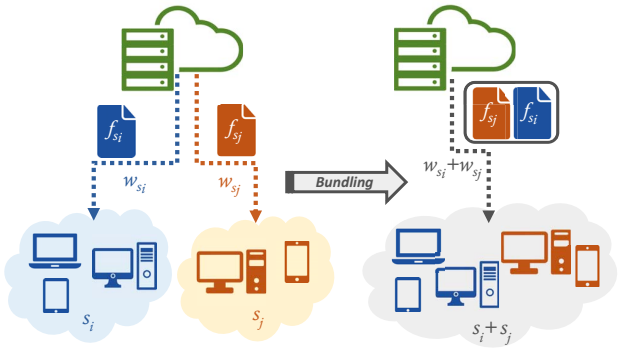



Fig. 2: Bundling scenario

### B. Cross-swarm bundling model

In this paper, we introduce *cross-swarm bundling* as the process of merging two swarms, say $s_i$ and $s_j$, together into a single swarm $s_i + s_j$ (see Fig.2). The files $f_{s_i}$ and $f_{s_j}$ are grouped into one file $f_{s_i} + f_{s_j}$ whose size is equal to the sum $F_{s_i} + F_{s_j}$. $f_{s_i} + f_{s_j}$ is sent via BitTorrent to the resulting swarm which contains all the peers from both $s_i$ and $s_j$. Once a client of a bundled swarm finishes downloading the original file and the required contribution, he can leave the swarm. The share of cloud bandwidth allocated to the bundle is equal to the sum of the shares allocated to each swarm individually before bundling.

To evaluate the efficiency of bundling on $s_i$ and $s_j$, we compare the download times before and after bundling. We denote by $t_{s_i}^{before}$ and $t_{s_j}^{before}$ the expected end download times for $s_i$ and $s_j$ respectively, before the application of bundling.

$t_{s_i}^{before}$ (resp. $t_{s_j}^{before}$) depends on the protocol being used by $s_i$ (resp. $s_j$). For example, if the peers in $s_i$ are using HTTP, then $t_{s_i}^{before} = t_{s_i}^{http}$ and it is calculated using (1). Otherwise, when the download protocol is BitTorrent, $t_{s_i}^{before} = t_{s_i}^{bt}$ as in (2). The same applies for $s_j$.

To this extent, we distinguish between three bundling variants, depending on the download protocol adopted by the grouped swarms: (i) *HTTP-swarm bundling*: when a HTTP swarm is grouped with another HTTP swarm, (ii) *BitTorrent-swarm bundling*: when a BitTorrent swarm is bundled with another BitTorrent swarm, and (iii) *Hybrid-swarm bundling*: when a HTTP swarm is bundled with a BitTorrent swarm.

$t_{s_i,s_j}^{after}$ is the expected end download time after the swarms are bundled. It is estimated using (2) since the download protocol adopted by the bundled swarms is always BitTorrent. The file size considered to calculate $t_{s_i,s_j}^{after}$ is $F_{s_i} + F_{s_j}$, where $F_{s_i}$ and $F_{s_j}$ are the amount of data that need to be downloaded for swarms $s_i$ and $s_j$ respectively. The cloud outgoing bandwidth allocated to the bundle is $w_{s_i} + w_{s_j}$, i.e. the aggregation of the bandwidth that were allocated to each swarm separately.

## V. Average case analysis: Bundling in personal clouds

From a personal cloud perspective, it is important to evaluate the efficiency of bundling before implementing a bundling mechanism. In this section, we focus on the case of bundling two HTTP swarms, the most common one in personal clouds. We propose the following notation related to the average characteristics of a swarm in the system:

- $\bar{L}$: average size of a swarm,
- $\bar{w}$: average cloud upload bandwidth allocated to a swarm,
- $\bar{F}$: average file size,
- $\bar{d}_{min}$: minimum download speed of the clients in an average swarm,
- $\bar{u}$: average upload speed of the clients in a swarm,
- $\bar{\eta}$: average value of the efficiency of file sharing,
- $\bar{t}^{before}$: average download time before bundling,
- $\bar{t}^{after}$: average download time after bundling.

Using this notation, we can estimate $\bar{t}^{before}$ and $\bar{t}^{after}$, using (1) and (2), as follows:

$$\bar{t}^{before} = \frac{\bar{F}}{\min\left\{\bar{d}_{min}, \frac{\bar{w}}{\bar{L}}\right\}}. \tag{3}$$

$$\bar{t}^{after} = \frac{2\bar{F}}{\min\left\{\bar{d}_{min}, \frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}}, 2\bar{w}\right\}}. \tag{4}$$

To compare both approaches, the service provider has to identify the bottleneck limiting the transfer of the files from the storage nodes to the end users. Using (3) and (4), we study all the possible values of $\min\left\{\bar{d}_{min}, \frac{\bar{w}}{\bar{L}}\right\}$ and $\min\left\{\bar{d}_{min}, \frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}}, 2\bar{w}\right\}$ and then compare $\bar{t}^{before}$ and $\bar{t}^{after}$ for each possible case of the bottleneck, as follows:

1) **Case 1:** $\bar{d}_{min} \leq \frac{\bar{w}}{\bar{L}}$ and $\bar{d}_{min} \leq \min\left\{\frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}}, 2\bar{w}\right\}$. *When the transfer bottleneck is the download speed of the peers, bundling is not efficient.*

   *Proof:*
   - $\bar{d}_{min} \leq \frac{\bar{w}}{\bar{L}} \Rightarrow \bar{t}^{before} = \frac{\bar{F}}{\bar{d}_{min}}$
   - $\bar{d}_{min} \leq \min\left\{\frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}}, 2\bar{w}\right\} \Rightarrow \bar{t}^{after} = \frac{2\bar{F}}{\bar{d}_{min}}$

   Since $\bar{F} > 0$ and $\bar{d}_{min} > 0 \Rightarrow \bar{t}^{after} > \bar{t}^{before}$. ■
2) **Case 2:** $\frac{\bar{w}}{\bar{L}} < \bar{d}_{min}$ and $2\bar{w} \leq \min\left\{\bar{d}_{min}, \frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}}\right\}$. *When the cloud's bandwidth is the bottleneck, bundling can be efficient if and only if the average swarm is composed of at least 2 peers.*

   *Proof:*
   - $\frac{\bar{w}}{\bar{L}} < \bar{d}_{min} \Rightarrow \bar{t}^{before} = \frac{\bar{F}\,\bar{L}}{\bar{w}}$
   - $2\bar{w} \leq \min\left\{\bar{d}_{min}, \frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}}\right\} \Rightarrow \bar{t}^{after} = \frac{\bar{F}}{\bar{w}}$

   It follows that $\left(\bar{t}^{before} > \bar{t}^{after} \Longleftrightarrow \bar{L} > 1\right)$. ■
3) **Case 3:** $\frac{\bar{w}}{\bar{L}} < \bar{d}_{min}$ and $\frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}} \leq \min\left\{\bar{d}_{min}, 2\bar{w}\right\}$. *When the upload capacity of the swarm is the bottleneck, bundling can be efficient if and only if the swarm's upload capacity is higher than the cloud's share of bandwidth allocated to the swarm.*

   *Proof:*
   - $\frac{\bar{w}}{\bar{L}} < \bar{d}_{min} \Rightarrow \bar{t}^{before} = \frac{\bar{F}\,\bar{L}}{\bar{w}}$
   - $\frac{\bar{w}+\bar{\eta}\bar{L}\bar{u}}{\bar{L}} \leq \min\left\{\bar{d}_{min}, 2\bar{w}\right\} \Rightarrow \bar{t}^{after} = \frac{2\bar{F}\,\bar{L}}{\bar{w}+\bar{\eta}\,\bar{L}\,\bar{u}}$

   Thus, $\left(\bar{t}^{before} > \bar{t}^{after} \Leftrightarrow \bar{\eta}\,\bar{L}\,\bar{u} > \bar{w}\right)$. ■

*Summary:* Based on this analysis, we can conclude that the efficiency of cross-swarm bundling in personal clouds depends on the transfer bottleneck. When the cloud has plenty of bandwidth to satisfy the demands of the clients, it is not worth bundling. However, when the outgoing bandwidth allocated to the personal cloud application is scarce, cross-swarm bundling presents a potential solution to reduce the download time for the clients.

## VI. Implementing bundling in personal clouds

In this section, we propose a mechanism to implement cross-swarm bundling in personal clouds. We start by calculating the gain in download time that will result from bundling. This gain represents the metric on which the bundling decision will be made. Later, we present a methodology to select the pairs of swarm to bundle based on graph matching techniques.

### A. Bundling metric: The expected gain in download time

The number of requests managed by the personal cloud can be relatively high especially at peak hours. This means that there are numerous bundling choices among the different swarms of clients. This makes the task of choosing the pairs[2] of swarms to be bundled a challenging one. Since the goal of this paper is to improve the QoE, we consider a bundling strategy based on the *expected gain in download time* that the peers in a given swarm $s_i$ will experience if $s_i$ is bundled with another swarm $s_j$. The idea is to estimate for each pair of swarms the expected gain in download time and based on the obtained values, group the pairs that would benefit the most.

$gain_{s_i}(s_j)$ measures the gain/loss that $s_i$ would experience if it is bundled with $s_j$. This gain can be defined as the normalized ratio of the difference between the expected end download times before and after bundling, as follows:

$$gain_{s_i}(s_j) = \frac{t^{before}_{s_i} - t^{after}_{s_i,s_j}}{t^{before}_{s_i}}. \tag{5}$$

$t^{before}_{s_i}$ and $t^{after}_{s_i,s_j}$ are calculated using (1) and (2), as explained in Section IV. $gain_{s_i}(s_j)$ gives an estimation of the gain from the perspective of the swarm $s_i$. To complete the picture, we present in (6) $gain_{s_i,s_j}$, which represents a weighted aggregation of the gain ratios of both swarms $s_i$ and $s_j$. The weights are allocated based on the number of clients in each swarm ($n_{s_i}$ and $n_{s_j}$).

$$gain_{s_i,s_j} = \frac{n_{s_i}\, gain_{s_i}(s_j) + n_{s_j}\, gain_{s_j}(s_i)}{n_{s_i} + n_{s_j}}. \tag{6}$$

$gain_{s_i,s_j}$ is positive if the bundling will result in a notable improvement in the download time for at least one of the

[2]To simplify the process of bundle selection, we only consider bundles of two swarms.

swarms. This is the case when both swarms would gain in download time or when the loss of one swarm is lower that the gain of the other. $gain_{s_i,s_j}$ is negative if the bundling will result in a notable degrade in the download time (at least for one swarm). This is the case when both swarms would lose in download time or when the gain of one swarm is lower than the loss of the other. It is important to note here that the gain is symmetric: $gain_{s_i,s_j} = gain_{s_j,s_i}$.

We consider now the set of all the swarms managed by the cloud and we suppose there is a total of $n$ swarms. To evaluate the efficiency of bundling for each pair of swarms, we calculate the matrix $Gain$. Each row in the matrix corresponds to a swarm $s_i$ and the columns represent the set of potential swarms to be bundled with. Each element in $Gain$ represents the gain, calculated using (6). Since $gain_{s_i,s_j} = gain_{s_j,s_i}$ and as it makes no sense to bundle the same swarm with itself, we only calculate the gains when $i < j$. Thus, $Gain$ is a strictly upper triangular matrix. $Gain$ is the objective matrix that contains all the expected gains (or losses) in the download times, for all the combinations of bundles. This matrix will serve to find the optimal set of swarms to bundle.

$$Gain = \begin{pmatrix} 0 & gain_{s_1,s_2} & gain_{s_1,s_3} & \cdots & gain_{s_1,s_n} \\ 0 & 0 & gain_{s_2,s_3} & \cdots & gain_{s_2,s_n} \\ 0 & 0 & 0 & \cdots & gain_{s_3,s_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}.$$

### B. Finding the optimal solution

The number of swarms managed by the personal cloud can be very high. This makes the task of choosing the right combination of swarms to bundle very challenging. In fact, for $n$ different swarms, the total number of possible combinations is: $\binom{n}{2} \times \binom{n-2}{2} \times \binom{n-4}{2} \ldots$.

From this large set of combinations, our goal is to find the "optimal" set of swarm pairs that *maximizes* the total sum of the gains. To do so, we follow the same strategy as Han et al. in [14]. We use the *maximum weight matching algorithm* [7], [9] to select the pairs of swarms that should be bundled together by converting the objective matrix $Gain$ into an undirected weighted graph $G = (V, E, w)$, where $w$ is the set of weights that are associated with the edges. Each vertex $v \in V$ represents a swarm. Each edge $e \in E$ connecting two swarms (say $s_i$ and $s_j$) signals the possibility to bundle these swarms together. The weight of the edge connecting $s_i$ and $s_j$ is equal to $gain_{s_i,s_j}$.

In graph theory, a *matching* is a subset of edges such that none of the selected edges share a common vertex. With respect to a weighted graph, a *maximum weight matching* is a matching for which the sum of the weights of the matched edges is as large as possible. The first polynomial time algorithm for maximum matching was proposed in 1965 by Edmonds [7] and subsequently improved by Gabow and others [8], [9]. Currently, there are several libraries that can find the maximum weight matchings for dense graphs in time $O(n^3)$ ($n$ is the number of nodes in the original graph). In our experimentation, we have noticed that this execution time does not exceed a few milliseconds on a regular desktop machine.

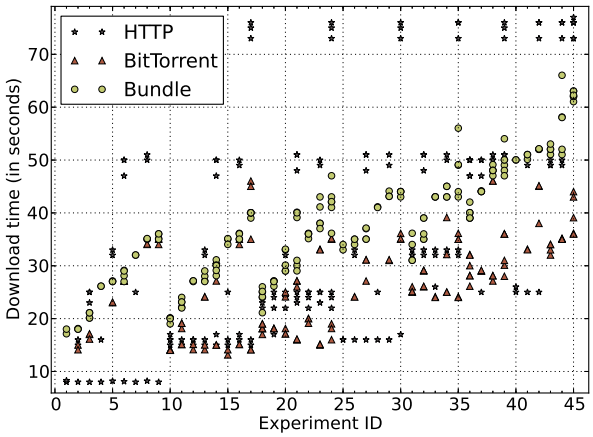



Fig. 3: Separate versus aggregated swarms: Measured download times for small swarms ($L_{s_i} \in \{1, 2, 3\}$) downloading small files ($F_{s_i} \in \{1, 2, 3\text{MB}\}$).

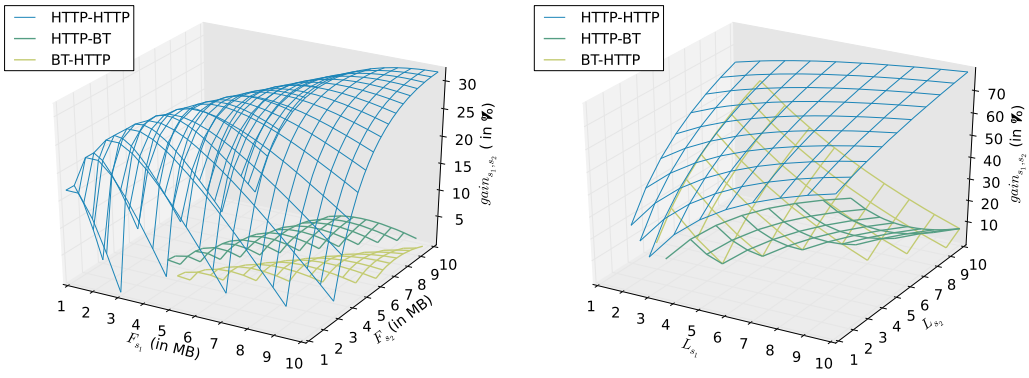
### C. Security concerns

Like most of the personal cloud systems, all the client-server communication in our system is over HTTPS, which protects against eavesdropping and tampering with the contents of the communication. To ensure data confidentiality when the transfer protocol is BitTorrent, we deploy a one-shot symmetric encryption mechanism that uses unique keys to encrypt the corresponding files.

To enforce security and privacy when the cloud decides to bundle two swarms $s_1$ and $s_2$, each of the corresponding files $f_1$ and $f_2$ is encrypted separately with a one-shot symmetric key, such as a DES or AES key. We denote by $k_1$ (resp. $k_2$) the key used to encrypt $f_1$ (resp. $f_2$). After encrypting $f_1$ and $f_2$, the cloud creates two metadata `.torrent` files. To ensure content delivery with BitTorrent, both torrents are sent to the peers in $s_1$ and $s_2$. However, the keys are only sent to the entities authorized to get the content: $k_1$ is only sent to the clients in $s_1$ and $k_2$ is only sent to the clients in $s_2$. The keys and the torrent files are sent to each of the requesters via HTTPS.

## VII. Validation

This section is dedicated to the validation of our bundling strategy. We start by proving, through experimentation, that cross-swarm bundling can reduce the download time for the clients. We focus in our experiments on small swarms and files which characterize personal clouds. After that, we apply our proposed bundling strategy on a trace of a real personal cloud system and prove that our approach can improve significantly the QoE without increasing the cloud's bandwidth consumption.




(a) $L_{s_1} = L_{s_2} = 2$ peers (b) $F_{s_1} = F_{s_2} = 1$ MB

Fig. 4: Estimated gain in download times for different combinations of swarms.

### A. *Bundling can reduce the download time*

The idea of bundling two different swarms in order to improve the QoE might seem controversial at first. To quantify the real effect of this approach, we ran experiments in a campus scenario and compared the measured download times of different peers with the two approaches: separate versus aggregated swarms.

*1) Experimental scenario:* We prepared a set of experiments[3] using all the combinations of swarms of sizes between 1 and 3 peers downloading small files: 1, 2 and 3 MB. The sizes of the swarms and the files are intentionally chosen to be small, in order to represent the typical swarms in a personal cloud [2], [3], [12]. We measure the download times for each of these swarms using HTTP and BitTorrent and then, for each combination of two swarms, we measure the download times when they are bundled together.

For HTTP transfers, we used the HFS[4]: HTTP File Server which is a file sharing server that supports bandwidth control. BitTorrent transfers are performed using the Vuze[5] Java BitTorrent client which is available under the GNU General Public License.

*2) Results:* Fig.3 represents the experimental results with all the download times measured for each experiment. We notice that the measured download times when swarms are bundled can be lower than with separate swarms, especially when both swarms were using HTTP before bundling. For the swarms which were already using BitTorrent, we notice that, in most cases, bundling does not improve the download time.

More concretely: we take the example of two identical swarms $s_1$ and $s_2$, each composed of 3 peers downloading a 3 MB file (Experiment 45 in Fig.3). When these swarms were downloading the files separately using HTTP (resp. BitTorrent), the measured download times for the peers in $s_1$ and $s_2$ ranged from 73 to 77 seconds (resp. 36 to 44 seconds). When $s_1$ and $s_2$ were bundled into a swarm composed of the same 6 peers downloading a 6 MB file, the experienced download time were between 61.05 and 63.051 seconds. This means that in the case of two HTTP swarms, all the peers experienced an improvement in download time higher than $10\%$ of the original time. However, when the swarms were using BitTorrent, bundling has resulted in a loss in download time of around $35\%$.

In addition to the improvement in download time with HTTP swarms, we notice that the use of bundling has reduced the amount of redundant data sent from the cloud servers. In fact, with 2 separate HTTP swarms, the cloud servers had to send the size of the file $f_1$ (resp. $f_2$) to each of the 3 peers in both $s_1$ (resp $s_2$). This means that the total amount of data sent by the cloud (without considering package losses) is equal to $3 \times 3 + 3 \times 3 = 18$ MB. When the swarms were bundled, the measured peers' contribution to the aggregated swarm was of 21,328 MB, which represents nearly $60\%$ of the total data to be sent (36 MB = 6 peers $\times$ 6 MB file). Compared to separate HTTP transfers, bundling reduced the load on the personal cloud and eliminated over $18\%$ of the redundant data transfers from the cloud to the end users.

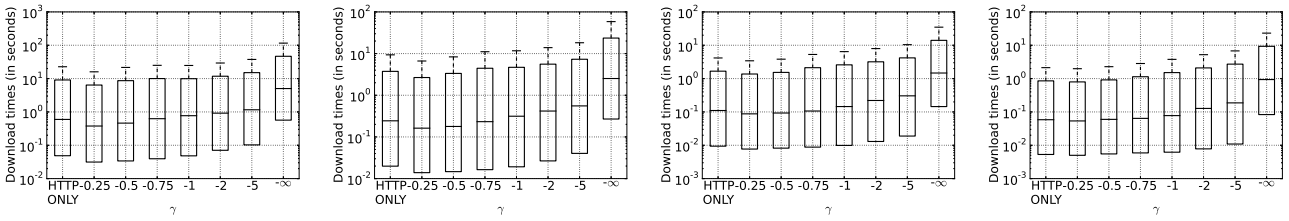
Fig.4 presents an overview of the combinations of swarms to bundle that can lead to an improvement in download time. Each swarm is defined by the number of peers and the size of the corresponding file. In each of the subfigures in Fig.4, we fixed one of these two parameters and evaluated the gain values for several combinations. In each combination, we calculated the gain using (6) for all possible download protocol combinations. We notice that the combinations where both swarms use HTTP before bundling lead to the highest improvements in download times. Nevertheless, we notice that even some HTTP-BitTorrent combinations can entail positive values of the gain when the file sizes (Fig.4a) and the swarm sizes (Fig.4b) get bigger.

*Summary:* Based on real experiments, we can confirm that cross-swarm bundling can reduce download time in this adverse scenario where files are small. But not only this, it

[3]The complete list of experiments and the obtained results can be found at http://ast-deim.urv.cat/lcn16/exps_lcn16.pdf

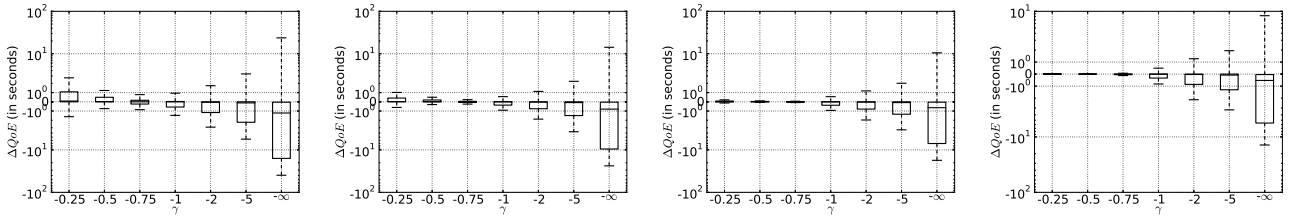
[4]HFS: HTTP File Server http://www.rejetto.com/hfs/

[5]Vuze: http://www.vuze.com/




(a) $W$ = 200 Mbps (b) $W$ = 250 Mbps (c) $W$ = 300 Mbps (d) $W$ = 350 Mbps

Fig. 5: Measured download times for different cloud bandwidth limits.




(a) $W$ = 200 Mbps (b) $W$ = 250 Mbps (c) $W$ = 300 Mbps (d) $W$ = 350 Mbps

Fig. 6: Measured $\Delta QoE$ for different cloud bandwidth limits.

helps reduce the outgoing bandwidth on the cloud side thanks to the contribution of peers. A key observation is that *bundling two HTTP swarms performs better than any other type of bundling* and that *the gain tends to increase when the sizes of the swarms and/or the files increase*.

### B. Evaluation bundling in a personal cloud

*1) The Ubuntu One trace:* In our validation, we use a real trace of the Ubuntu One (UB1) system. The trace was provided by *Canonical Ltd.*[6] in the context of the CloudSpaces[7] project. The logs were collected for about a month from their servers located in London, based on the behavior of real users. A complete analysis of the trace is provided in [12].
In this paper, we use a sample[8] of the trace that corresponds to an hour during a random day. The sample accounts for $225,514$ download operations related to $173,756$ distinct files. The average file size in the sample is about 1 MB, which can be viewed as a *worst case* for cloud content distribution. For each download operation, several information was collected, including: the timestamp, the hash and size of the file in question, the user hash identifier and the corresponding upload and download bandwidths. For the sake of privacy, files and user real identifiers were presented in the form of unique hash codes.

*2) Experimental settings:* To evaluate the efficiency of our proposal, we developed a Python[9] simulator that uses the UB1 trace and re-simulates the arrival pattern of the clients. The simulator implements the following strategies:

*a) HTTP_ONLY:* This strategy represents the default behavior of a personal cloud where HTTP is the only download protocol used to distribute the files from the cloud servers to the end users.

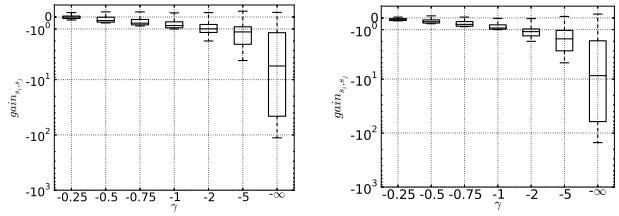
*b) MAX_GAIN:* This scenario corresponds to the application of bundling across different swarms of clients. At each timestamp, the objective matrix $Gain$ is calculated. The choice of swarms to bundle is made based on the application of the maximum weight matching algorithm.

*c) MAX_GAIN> $\gamma$:* This scenario is similar to the previous one. However, it adds an extra constraint for the swarms to be bundled in order to avoid taking bundling decisions that could lead to notable degrades in performance. To this extent, the objective matrix $Gain$ is filtered before being converted to a graph, and only the combinations $(s_i, s_j)$ that satisfy the condition $gain_{s_i,s_j} > \gamma$ are considered. $\gamma$ is a real value which can be set by the cloud provider. $\gamma$ can be positive or negative based on the current load on the cloud. A negative value of $\gamma$ means that losses of up to $\gamma$ times of the original download time are tolerated. For instance, $\gamma = -1$ means that the bundles considered are the ones that can lead to an increase in download time equal to the original download time at most. Based on our experiments, we have noticed that putting such a limit is very important to prevent degrading the download time for the clients. Note that the *MAX_GAIN* scenario corresponds also to the case *MAX_GAIN*> $\gamma$ where $\gamma = -\infty$.

In our implementation of the simulator, we use the Networkx[10][13] graph matching library. Networkx is based on

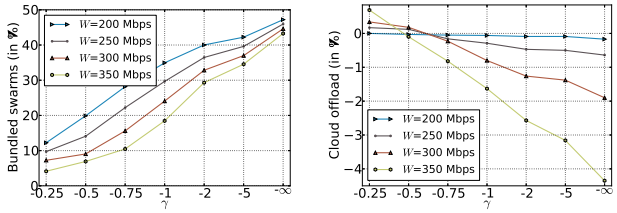
[6]Canonical Ltd. http://www.canonical.com
[7]The CloudSpaces project: http://cloudspaces.eu
[8]The sample is available at: http://ast-deim.urv.cat/lcn16/ub1_sample.txt
[9]Python Software Foundation: http://www.python.org

[10]NetworkX is a Python software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. Networkx is available under the BSD license at: https://github.com/networkx




(a) $W = 200$ Mbps (b) $W = 350$ Mbps

Fig. 7: Values of the bundling metric $gain_{s_i,s_j}$ at the moment of the bundling decision.




(a) Bundled swarms percentage (b) Cloud offload percentage

Fig. 8: Percentages of cloud offload and bundled swarms

the "blossom" method for finding augmenting paths and the "primal-dual" method for finding a matching of maximum weight, both methods invented by Edmonds [9].

*3) Results:* We exploit the previously described trace sample of UB1 and re-simulate the arrival pattern of the peers to validate our approach. We run our simulator with different limits $\gamma \in \{-0.25, -0.5, -0.75, -1, -2, -5, -\infty\}$. We collect the logs of each experiment and evaluate our proposal comparing the results with the ones obtained using HTTP alone.

Fig.5 presents the download times for all the download operations in the trace for different cloud bandwidth limits $W \in \{200, 250, 300, 350$ Mbps$\}$. Compared with the HTTP_ONLY scenario, we notice that bundling performs better under smaller bandwidth limits. Surely, when the $W$ gets higher, the download times are shorter and the overall performance of the system improves. However, in these cases, bundling does not improve the download times and sometimes it can even degrade the performance of the system. This result agrees very well with our analysis in Section V.

Fig.7 shows the importance of the $\gamma$ constant. It depicts a box plot of the values of the bundling metric $gain_{s_i,s_j}$ for each couple $(s_i, s_j)$ at the moment of bundling. When $\gamma = -\infty$ (MAX_GAIN scenario), the bundling combinations output by the graph matching library can lead to important degradation in download time. In fact, the expected $gain_{s_i,s_j}$ in that scenario could reach -$10^2$ which reflects a resulting download 100 times longer than the original one. This proves that setting $\gamma$ to a reasonable value is very important to prevent degrading the download time for the clients. This importance can be further perceived in Fig.6. This figure presents the $\Delta QoE$ for all the download operations in the trace for different cloud bandwidth limits. $\Delta QoE$ is calculated for each download operation as the difference between the download time measured with HTTP only and the one using bundling. We remind that a positive value of $\Delta QoE$ reflects an improvement in download time compared to the use of HTTP only, while a negative $\Delta QoE$ means that the use of bundling has affected negatively the download time and made it longer. We notice that when bundling was deployed with $\gamma = -\infty$, most of the clients have experienced an increase in download time of about 10 seconds. Nevertheless, when $\gamma$ is small, the majority of the operations have witnessed an improvement in download time of a few seconds.

Fig.8a presents the percentage of bundled swarms as a function of $\gamma$. We notice that the number of bundled swarms increases with $\gamma$. As a matter of fact, when $\gamma$ is small, the bundling combinations that satisfy the condition $gain_{s_i,s_j} > \gamma$ are more limited which explains the limited percentage of bundled swarms.

In addition to the improvement in download time, the use of bundling does not increase the usage of the cloud's bandwidth. Fig.8b depicts the cloud offload percentages for different combinations of $\gamma$ and $W$. In accordance with the analysis in Section V, we notice savings in cloud bandwidth utilization when the $\Delta QoE$ is positive.

*Summary:* Based on the UB1 trace, we confirm that *cross-swarm bundling can be useful to improve QoE in personal clouds*, especially when the available outgoing bandwidth is small. In this case, cross-swarm bundling has a positive side effect by reducing the server side bandwidth requirements. The study of the trace has also proven the importance of *filtering the objective matrix before applying graph matching* in order to avoid taking bundling decisions that could lead to degrades in the QoE.

## VIII. Conclusions

In this paper, we propose to use cross-swarm bundling in order to improve the QoE for the clients by leveraging their spare upload capacities. To this extent, we present a methodology to implement this feature in personal clouds. Our proposal is validated using a real trace of Ubuntu One. The results prove the efficiency of the approach in improving the QoE for the clients compared to the classic distribution methods.

Bundling can be adopted by personal clouds to improve their performance and gain more clients. Our future plans include the study and evaluation of new bundling metrics. We plan also to extend the idea of bundling to systems (other than personal clouds) with bigger shared files.

## Acknowledgments

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness in the context of the project

*Cloud Services and Community Networks* (TIN2013-47245-C2-2-R) and by EU in the context of the projects *CloudSpaces* (FP7-317555) and *IOStack* (H2020-ICT-2014-7-1).

## REFERENCES


[1] R. Chaabouni, P. Garcia-Lopez, M. Sanchez-Artigas, S. Ferrer-Celma, and C. Cebrian, "Boosting content delivery with BitTorrent in online cloud storage services," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, Sept 2013, pp. 1–2.

[2] R. Chaabouni, M. Sanchez-Artigas, and P. Garcia-Lopez, "Reducing costs in the personal cloud: Is bittorrent a better bet?" in *Peer-to-Peer Computing (P2P), 14-th IEEE International Conference on*, Sept 2014, pp. 1–10.

[3] R. Chaabouni, M. Sanchez-Artigas, P. Garcia-Lopez, and L. Pamies-Juarez, "The power of swarming in personal clouds under bandwidth budget," *Journal of Network and Computer Applications*, vol. 65, pp. 48 – 71, 2016.

[4] B. Cohen, "Incentives Build Robustness in BitTorrent," 2003.

[5] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking Personal Cloud Storage," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC'13. ACM, 2013, pp. 205–212.

[6] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside Dropbox: Understanding Personal Cloud Storage Services," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. ACM, 2012, pp. 481–494.

[7] J. Edmonds, "Paths, trees and flowers," *Canadian Journal of Mathematics*, vol. 17, pp. 449–467, 1965.

[8] H. N. Gabow, "Implementations of algorithms for maximum matching on nonbipartite graphs," Ph.D. dissertation, Stanford University, 1973.

[9] Z. Galil, "Efficient Algorithms for Finding Maximum Matching in Graphs," *ACM Comput. Surv.*, vol. 18, no. 1, pp. 23–38, Mar. 1986.

[10] P. Garcia-Lopez, M. Sanchez-Artigas, C. Cotes, G. Guerrero, A. Moreno, and S. Toda, "StackSync: Architecturing the Personal Cloud to Be in Sync." [Online]. Available: http://stacksync.org/wp-content/uploads/2013/11/stacksync_full_paper.pdf

[11] P. Garcia-Lopez, M. Sanchez-Artigas, S. Toda, C. Cotes, and J. Lenton, "Stacksync: Bringing elasticity to dropbox-like file synchronization," in *Proceedings of the 15th International Middleware Conference*, ser. Middleware '14. ACM, 2014, pp. 49–60.

[12] R. Gracia-Tinedo, Y. Tian, J. Sampe, H. Harkous, J. Lenton, P. Garcia-Lopez, M. Sanchez-Artigas, and M. Vukolic, "Dissecting ubuntuone: Autopsy of a global-scale personal cloud back-end," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, ser. IMC '15. ACM, 2015, pp. 155–168.

[13] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., 2008, pp. 11 – 15.

[14] J. Han, T. Chung, S. Kim, H. chul Kim, J. Kangasharju, T. T. Kwon, and Y. Choi, "Strategic bundling for content availability and fast distributionin BitTorrent," *Computer Communications*, vol. 43, no. 0, pp. 64 – 73, 2014.

[15] J. Han, T. Chung, S. Kim, T. T. Kwon, H.-c. Kim, and Y. Choi, "How Prevalent is Content Bundling in BitTorrent," in *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '11. ACM, 2011, pp. 127–128.

[16] J. Han, S. Kim, T. Chung, T. T. Kwon, H.-c. Kim, and Y. Choi, "Bundling Practice in BitTorrent: What, How, and Why," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '12. ACM, 2012, pp. 77–88.

[17] R. Kumar and K. Ross, "Peer-Assisted File Distribution: The Minimum Distribution Time," in *Hot Topics in Web Systems and Technologies, 2006. HOTWEB '06. 1st IEEE Workshop on*, Nov 2006, pp. 1–11.

[18] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai, and Z.-L. Zhang, "Towards network-level efficiency for cloud storage services," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. ACM, 2014, pp. 115–128.

[19] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Y. Zhao, C. Jin, Z.-L. Zhang, and Y. Dai, "Efficient batched synchronization in dropbox-like cloud storage services," in *Proceedings of the 14th International Middleware Conference*, ser. Middleware '13, 2013, pp. 307–327.

[20] D. S. Menasche, A. A. Rocha, B. Li, D. Towsley, and A. Venkataramani, "Content Availability and Bundling in Swarming Systems," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. ACM, 2009, pp. 121–132.

[21] S. Zhang, N. Carlsson, D. Eager, Z. Li, and A. Mahanti, "Dynamic File Bundling for Large-scale Content Distribution," in *Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012)*, ser. LCN '12. IEEE Computer Society, 2012, pp. 601–609.

[22] S. Zhang, N. Carlsson, D. L. Eager, Z. Li, and A. Mahanti, "Towards a Dynamic File Bundling System for Large-Scale Content Distribution," in *MASCOTS 2011, 19th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Singapore, 25-27 July, 2011*, 2011, pp. 472–474.