Efficient peer-to-peer backup services through buffering at the edge

Serge Defrance, Anne-Marie Kermarrec, Erwan Le Merrer, Nicolas Le Scouarnec, Gilles Straub, Alexandre van Kempen

Efficient peer-to-peer backup services through buffering at the edge

Serge Defrance*, Anne-Marie Kermarrec†, Erwan Le Merrer*,
Nicolas Le Scouarnec*, Gilles Straub* and Alexandre Van Kempen*
*Technicolor, Rennes, France
†INRIA Rennes, Bretagne Atlantique, France

***Abstract*—The availability of end devices of peer-to-peer storage and backup systems has been shown critical for usability and for system reliability in practice. This has led to the adoption of hybrid architectures composed of both peers and servers. Such architectures mask the instability of peers thus approaching the performances of client-server systems while providing scalability at a low cost. In this paper, we advocate the replacement of such servers by a cloud of residential gateways, as they are already present in users' homes, thus pushing the required stable components at the edge of the network. In our gateway-assisted system, gateways act as buffers between peers, compensating for their intrinsic instability. This enables to offload backup tasks quickly from the user's machine to the gateway, while significantly lowering the retrieval time of backed up data. We evaluate our proposal using real world traces including existing traces from Skype and Jabber as well as a trace of residential gateways for availability, and a residential broadband trace for bandwidth. Results show that the time required to backup data in the network is comparable to a server-assisted approach, while substantially improving the time to restore data, which drops from a few days to a few hours. As gateways are becoming increasingly powerful in order to enable new services, we expect such a proposal to be leveraged on a short term basis.**

I. Introduction

While digital data clearly dominates, backup is of the utmost importance. More specifically, online (*i.e.* off-site) backup is often preferred over simple backup on external devices as it ensures data persistence regardless of the damage cause (*e.g.* failures, burglars or even fires). To enable their deployment, online backup systems should run in the background and provide reasonable performances so that archives can be stored safely in reasonable times. While cloud backup systems are increasingly adopted by users (*e.g.* Amazon S3 or DropBox), their peer-to-peer alternatives, potentially offering *virtually unlimited storage* for backup [1], [2], are still not appealing enough performance-wise, as *e.g.* retrieval times for saved data can be an order of magnitude higher that the time required for direct download [3].

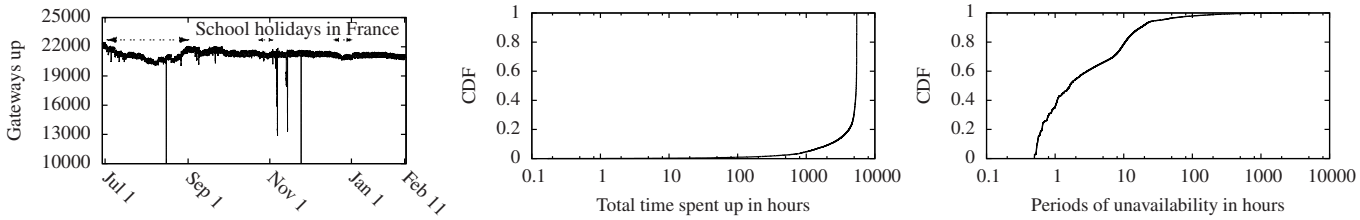
Indeed, peer-to-peer backup systems are limited by the low to medium availabilities of participating peers and by the slow up-links of peers' network connections. This limits the amount of data that peers can transfer and places peer-to-peer systems way behind datacenter-based systems [4]. Not only this may impact the reliability of the stored content but also this does not provide a convenient system for users. We focus on this problem and investigate a new way of performing efficient backup on commodity hardware in a fully peer-to-peer way.

In this paper, we propose a new architecture for peer-to-peer backup, where residential gateways are turned into a *stable buffering layer* in between the peers and the Internet. The residential gateways are ideal to act as stable buffers: they lay at the edge of the network between the home network and the Internet, and are highly available since they remain powered-on most of the time [5]. Our approach relies on the high availability of gateways to compensate for the end peers instability. Our approach enhances the backup system's performance along two lines:

- The network connection can be used more efficiently (*i.e.* the available bandwidth can be exploited typically 21h/day instead of only 6 to 12h/day on average). This leads to significant enhancements. For example, we observe that the time to backup a 1GB archive is reduced from around one week in a pure peer-to-peer system to around one day in our system.
- Additionally, the gateways, offering a high availability (86% on average, according to our measurements), can act as rendezvous to allow any two peers to communicate efficiently, even if they are not up at the same time. In our application, this enhancement mainly has an impact on the time to restore, which is reduced from a few days to a few hours.

Our proposal differs from existing approaches [4], [6]–[10] by taking into account the low-level structure of the network. Indeed, most peer-to-peer applications ignore the presence of a gateway in between each peer and the Internet. As a result they do not leverage the presence of the gateway while it can greatly improve the overall performance of the system. We believe that leveraging the gateway storage space may render peer-to-peer systems viable alternatives for backup. This should provide a reasonable solution even when peers experience a low availability as long as they connect frequently enough to the system. Using those gateways as buffers between peers participating in a backup or restore operation, enables to implement a stable rendezvous point between transient peers.

The remainder of this paper is structured as follows. In Section II, we briefly review some pieces of work that motivated our proposal. In Section III, we detail our architecture and sketch the storage system. Section IV introduces a framework for comparison of our proposal to that of competitors, and Section V presents our evaluation study. We discuss respec-



(a) Availability VS time (b) CDF of Total uptime (c) CDF of Dowtime duration

Figure 1. Availability of residential gateways mesured on a French ISP. The dataset has been acquired sending pings to a random sample of gateway IPs.

tively some specific points and related work in Section VI and Section VII. Finally, we conclude the paper in Section VIII.

II. Background

Peer-to-peer storage systems initially relied on the set of all participating peers, typically constituted of users' desktop PCs, without any further infrastructure [6], [7]. However, it has been acknowledged since then [11], [12] that those pure peer-to-peer architectures may fail to deliver reliable storage by exploiting the resources of peers, mainly due to the low availability of peers and the slow up-link of their network connections. One straightforward solution is to exclude peers with a low availability or a slow network connection to access the service [13]; this nevertheless excludes many participants and significant amounts of exploitable resources [1], [2].

In order to move towards practical system deployment while still leveraging users' resources, hybrid architectures, where both servers and peers coexist, have been very recently proposed in various contexts [14]. The problem of sharing files while mitigating the load of central servers is addressed in [15], which proposes a BitTorrent like server-assisted architecture where central servers act as permanently available seeders. Lastly, a server assisted peer-to-peer backup system is described in [4]. In their system, which can be referred to as *CDN-assisted*, the CDN enables to reduce the time needed to backup data, while the use of peers guarantees that the burden of storage and communication on the data center remains low. In this last approach, a peer uploads data to a set of other peers if they are available, and falls back on the datacenter otherwise, thus using the datacenter as a stable storage provider.

Residential gateways connect home local area networks (LAN) to the Internet. They act as routers between their WAN interface (Cable or DSL) and their LAN interfaces (Ethernet and WiFi). They started to be deployed in homes to share Internet access among different devices and got additional functions as services (VoIP, IPTV) were delivered over the Internet. It is now fairly common to have home gateways embedding a hard drive, acting as Network Attached Storage to provide storage services to other home devices and offering some other ones to the outside world [5], [16].

III. A gateway assisted system

A. Stability of residential gateways

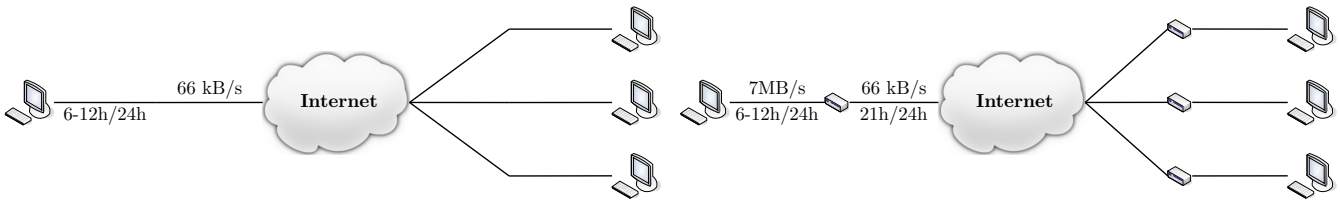
As residential gateways provide not only Internet connectivity, but also often VoIP, IPTV and other services to the home, the intuition tells us that they remain permanently powered on. To confirm this assumption, we extracted a trace of residential gateways of the French ISP Free, using active measurements[1]. We periodically ping-ed a set of IP addresses randomly chosen in the address range of this ISP, which has a static IP addressing scheme. We obtained the uptime patterns of $25,000$ gateways for 7.5 months, covering week-patterns [17], [18], and holidays. We plot the availability of those devices against time, in the classical representation of availability, on Figure 1a. Some clear acquisition artifacts appear due to both the unreliability of the ICMP monitoring and temporary failures on the path between our platform and the targeted network. Yet, the trace confirms the common intuition about the stability of those devices, in spite of a few users having power-off habits (on a daily or a holiday basis, see Figure 1c), thus slightly reducing the average availability. The average availability of gateways in this trace is 86%, which confirms the results observed in [5], where the authors used traces from a biased sample (only BitTorrent users) [19]. This has to be contrasted with the low to medium availabilities of peers generally recorded in the literature, as *e.g.* 27% in [4], or 50% in [20].

This increased stability makes gateways appealing candidates for backing peer-to-peer services. The intuition is that data is temporarily stored on gateways to compensate for peers transient availability. In this paper, we advocate the use of gateways as buffers and not storage. This choice is motivated by the increasing number of devices embedding storage, within the home and attached to a gateway. Dimensioning the storage of the gateway accordingly would be costly and would break the peer-to-peer paradigm by creating a central point in charge of hosting resources of attached devices durably: the contributed resources would no longer scale with the number of clients. In our buffer model, each device is required to provide a portion of its available space [1], [2], to participate to the global backup system.

B. System rationale

In this paper, we propose to decentralize the buffer logic implemented in [4] by a CDN, in order to provide a reliable backup system despite the dynamic nature of peers composing

[1]This trace and additional information can be found at the following URL http://www.thlab.net/~lemerrere/trace_gateways/ . To the best of our knowledge, this is the first trace tracking availability of gateways.



(a) With passive gateways

(b) With active gateways

Figure 2. A global picture of the network connecting the peers to the service. Those end-devices are available $6-12$h/day. If we allow the gateway, which is available 21h/day, to perform buffering, we can benefit from the speed difference between local links (7MB/s) and ADSL links (66KB/s).

the network. Our system is specifically tailored for the current architecture of residential Internet access. Indeed, most previous works assume that peers are directly connected to the network (see Figure 2a) while, in most deployments, a residential gateway is inserted in between the peers in the home network and the Internet. Hence, a realistic low-level network structure is composed of *(i)* peers, connected to the gateway through Ethernet or Wifi, *(ii)* residential gateways, providing the connection to the Internet, and *(iii)* the Internet, which we assume to be over provisioned (architecture depicted on Figure 2b). In our approach, we propose to use storage resources of residential gateways, thus creating a highly available and distributed buffer to be be coupled with peers.

Such an architecture is appealing as it takes into account *(i)* the availability that differs between peers and gateways, and *(ii)* the bandwidth that differs between the LAN and the Internet connection. Firstly, the peers tend to have a low to medium availability (*i.e.* from 25% or 6 hours/day on average on a Jabber trace, to 50% on a Skype trace we introduce later on) while gateways have a high availability (i.e., 86% or 21 hours/day on average). Secondly, peers are connected to the gateways through a fast network (at least 7MB/s) while the Internet connection (between gateways and the Internet) is fairly slow (*i.e.* 66 kB/s on average for ADSL or Cable). Our architecture exploits the major difference of throughput between the LAN and the Internet connection (WAN) by offloading tasks from the peer to the corresponding gateway quickly, thus using the Internet connection more efficiently (*i.e.* 21h/day instead of only $6-12$h/day on average).

This enables the large-scale deployment of online storage applications by fixing the issues provoked by the combination of slow up-links and short connection periods (as in the case of pure peer-to-peer). These issues are becoming increasingly important as the size of the content to backup increases while ADSL bandwidth has not evolved significantly over the past years. For example, uploading 1GB (a 300 photo album) to online storage requires at least 4h30 of continuous uptime. Hence, these applications require users to change their behavior (*e.g* let their computers powered for the whole night to be able to upload large archives); this limits their deployment and makes automated and seamless backup close to impossible. Our approach precisely aims at combining peers' fast but transient connections with gateways' slow but permanent connections. Following this logic, if peers upload directly to the Internet, they can upload on average 1.4-2.8GB/day (Fig. 2a); if we consider that the gateway is an active equipment that can perform buffering, a peer can upload 148-296GB/day to the gateway and the gateway can upload on average 4.8GB/day (Fig. 2b). We then advocate that turning the gateway into an active device can significantly enhance online storage services, be they peer-to-peer or cloud systems.

In the last part of this section, we propose the design of a gateway-assisted peer-to-peer storage system (*GWA*) based on these observations, and relying on two entities: *(i)* users' gateways, present in homes and providing Internet connectivity, and *(ii)* peers, being users' devices connected to the Internet (through a gateway) and having some spare resources to contribute to the storage system.

C. Gateway-assisted storage system

We consider a general setting to backup data to third parties on the Internet, generic enough for us to compare approaches from related work in the same framework.

The content to be backed up is assumed to be ciphered prior to its introduction in the system, for privacy concerns. The content can be located in the distributed storage system through an index, which can be maintained, for example by a *distributed hash table* connecting each piece of content stored to the set of peers hosting it. We consider that users upload data from one peer, under the form of archives. In order to achieve a sufficient reliability, the system adds redundancy to the content stored. To this end, it splits the archive into k blocks, and adds redundancy by expanding this set of k blocks into a bigger set of n blocks using erasure correcting codes [21] so that any subset of k out of n blocks allows recovering the original archive. This enables to increase the file availability as the resulting system-wide availability is

$$A = \sum_{i=k}^{n} \binom{n}{i} \bar{p}^i (1-\bar{p})^{n-i} \tag{1}$$

where $\bar{p}$ is the average availability of peers, which is smaller than A. In the rest of this paper, we set a target A_t for the system-wide availability so that n must be the smallest n ensuring that $A > A_t$. Intuitively, the availability targeted by the application is the portion of time a backed up data is online for restore. High availability rates have been shown cumbersome to reach in dynamic systems [11], so a reasonable trade-off should be considered [3].

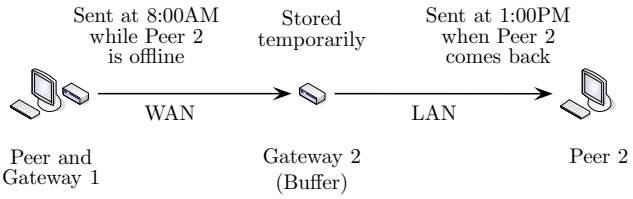


Figure 3. Backup operation: buffering a block at a random gateway

For a backup operation, the client peer uploads the file and the redundancy blocks to other peers as follows:

1. Prepare. As soon as it gets connected, the client peer starts pushing the archive at LAN speed to its gateway, which buffers the data. At this point, the data has been partially backed up but the final level of reliability is not yet guaranteed.

2. Backup. In our system, the gateway is in charge of adding the redundancy; this allows faster transfer from the peer to the gateway as a lower volume of data is concerned. Once done, it starts uploading data to other gateways, at WAN speed (left-hand side of Figure 3). Gateways are active devices that can serve peer requests thus ensuring data availability and durability even if data is not fully pushed to remote peers. Therefore, data can be considered totally backed up when all blocks have reached the selected set of independent remote gateways.

3. Offload. Finally, remote gateways offload, at LAN speed, the content to their attached peers (right-hand side on Figure 3) as soon as the attached peer becomes available.

A user can request access to its data at anytime; the success of immediate data transfer from the storage system to the requesting peer depends on the targeted availability of the backup, that has been set by the system administrator. To reclaim backed up data, the role of all elements in the systems are reversed and the restore is performed as follows:

1. Fetch. To access a data, the requesting client peer informs its gateway of the blocks it is interested in. The client gateway carries on the download on behalf of the client peer by contacting the remote gateways handling peers where the data was uploaded. If the data was offloaded to some peer, it is fetched as soon as possible by the corresponding remote gateway.

2. Restore. Then the remote gateway sends the data to the requesting client gateway.

3. Retrieve When the client gateway has succeeded in getting the whole content (the data has been restored), it informs the client peer that its retrieval request has been completed, as soon as it connects back.

IV. A COMPARISON FRAMEWORK FOR BACKUP SCHEMES

We extensively evaluate our approach through simulations, taking as inputs execution traces of large-scale deployed systems. This section first presents the experimental setup, the competing approaches, namely pure peer-to-peer (noted *P2P* hereafter) and CDN-assisted (noted *CDNA*), and the performance metrics.

A. Parameters and data sets

The setting we described in previous section comes with the following set of parameters:

Network Bandwidth: On the WAN side (Internet connection), the bandwidth is heterogeneous and set according to the traces from a study of residential broadband networks [19]; it exhibits an average of 66 kB/s[2]. On the LAN side, we assume a constant bandwidth of 7 MB/s, capturing both wired and WiFi connections in home environments. The LAN bandwidth only impacts our gateway-assisted approach, as other approaches suffer from the bottleneck at the WAN interface since they upload data directly to devices in the WAN (peers or CDN).

Backup and Restore requests: The backup and restore user requests are modeled with Poisson processes, which are known to match user requests [20], of parameters λ_{backup} and λ_{restore} that represent the rate of backup and restore requests in the system. Each request is related to one archive. We assume that all archives have the same fixed size for all peers, typically representative of some large multimedia content (we make this size vary). As a result, the behavior of peers is assumed to be homogeneous: we are not aware of any trace giving hints about the real behavior of users within peer-to-peer storage/backup systems.

Peer availability: In order to model the up-time of personal computing devices (*i.e.* peers), we rely on two instant messaging traces. We only remove from the traces the peers with an availability lower than 1%, since these have a behavior not compatible with a backup application running in the background (*e.g.* people that have tried Skype only once but never use it again).

- **Skype** In this trace [20] of about 1269 peers over 28 days, the average availability of peers is 50%, which represents a medium availability when considering peer-to-peer systems.
- **Jabber** In this trace, provided by the authors of [4], the average availability is 27%. We used a slice of 28 days containing 465 peers.

All peers strictly follow the behavior of the trace. In particular, since our backup application runs in the background, *client* peers are not assumed to remain connected during the whole backup (their presence follows the IM/Skype trace), as opposed to the assumptions made in [4]. Note that these traces may however over-estimate uptime period since people using Skype are likely to be online more often than the average user.

Gateway availability: To model the up-time of residential gateways, we rely on our gateway trace presented in Section III. Since the gateway and peer traces have been obtained independently, they do not capture the correlation between the behavior of a peer and of the associated gateway. Hence, we randomly assign a gateway to each peer. In order to avoid unrealistic scenarios where the peer is up while the gateway is down, we assume a gateway to be available when one of its attached peers is up, to allow communication between them:

[2]Note that all amounts of data and bandwidth are given in bytes.

we rely on the gateway trace only for gateway to gateway communication.

Redundancy Policy: As explained previously, the redundancy policy is based on erasure correcting codes and is entirely determined by the number of blocks k each archive is split into and by the targeted availability A_t, which is set by the administrator. The backup is thus considered as complete when there are enough redundancy blocks n backed up in the network to guarantee a system-wide availability of at least A_t. The relation between the availability of peers and the values k and A_t has been studied in papers studying in details redundancy policies [11], [21], [22].

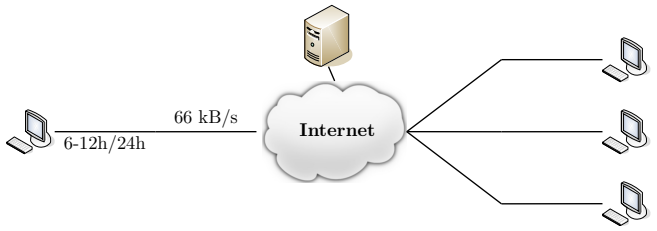


Figure 4. A global picture of the network connecting peers and CDN, as used in [4]. Note that the CDN (Server) has an infinite capacity and 100% availability. However, since the bandwidth used at the CDN is billed, the CDN is not used as a *relay* but as another kind of end-storage (*i.e.* it does not upload content to other peers but only stores content temporarily until the backing up peers have uploaded content to enough peers.)

B. Competitors

We compare the performance of our *GWA* scheme against the two main classes of related backup systems, namely *P2P* and *CDN Assisted*, within the same simulation framework.

P2P The vast majority of peer-to-peer storage protocols historically presents a purely decentralized system with one-to-one uploads/downloads, without servers [6], [7]. They assume that gateways are passive devices that cannot store and forward but only route packets. This protocol is similar to the protocol we described in the previous section but does not have active gateways acting as buffers.

CDNA A possible enhancement consists in introducing a CDN to mitigate the low availability of peers [4]. The CDN is a central service in the core network, having unbounded capacity. We consider the most peer-to-peer variant of the protocol in [4] (*i.e.* the opportunistic one). In this protocol, the peers upload content to other peers in priority and upload to the CDN only when the whole bandwidth is not used (*i.e.*, not enough remote peers are available). This enables to lower time to backup by avoiding waiting times. However, the CDN does not upload the content to remote peers, but client peers eventually upload again to remote peers thus uploading twice in some cases. Indeed, pricing schemes at CDN implies that uploading from the CDN should be avoided so as to reduce costs. A schematical view is given in Figure 4. The CDN never fails, hence, a single copy of the content on CDN is enough to ensure an availability of 100%. As a result, a data backup is successful as soon as s fragments have been uploaded to the storage server and t fragments have been uploaded to the peers so that the targeted availability is guaranteed, as stated in (2).

$$\sum_{i=k-s}^{t} \binom{t}{i} \bar{p}^i (1-\bar{p})^{t-i} > A_t \tag{2}$$

$\bar{p}$ is the average availability of a peer and A_t is the targeted availability.

C. Simulation setup & Performance Metrics

We implement a *cycle-based* simulator to compare our *GWA* architecture with the two aforementioned alternatives. Depending on the simulated architecture namely *P2P*, *GWA* or *CDNA*, the system is respectively composed of only peers, peers and gateways, or peers and a CDN. Devices transfer their data depending on their upload bandwidth and on their behavior, which are *trace-driven* for peers and gateways while CDN is always up. We evaluate the time for each backup or restore request to be completed, while measuring the storage needs of the system.

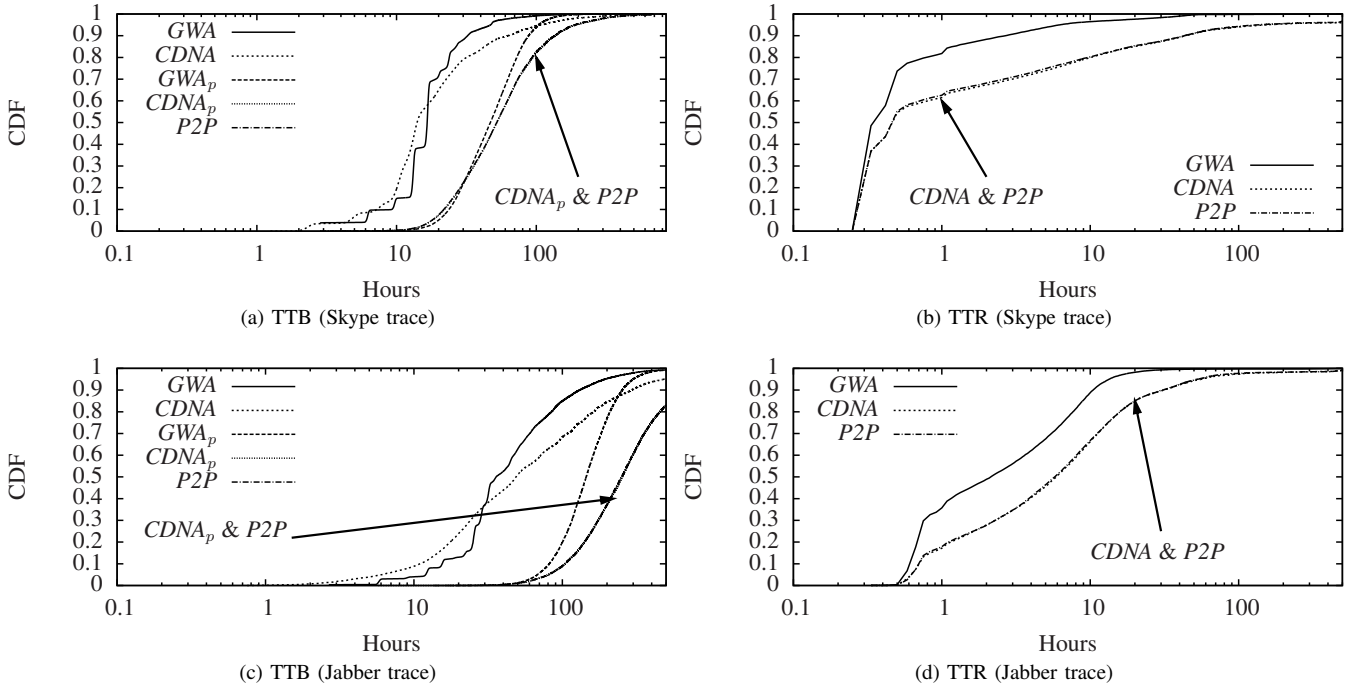
In our simulations, the redundancy policy relies on the following values $k_{skype} = 16$, $k_{jabber} = 8$, $A_t = 0.7$, $\bar{p}_{skype} = 0.5$ and $\bar{p}_{jabber} = 0.27$ leading to $n_{\text{skype}} = 35$ and $n_{\text{jabber}} = 33$. We assign to each peer a set of n distinct remote peers in a uniform random way, for example using the hashing scheme of a distributed hash table. When a given peer requests a data backup operation, the n encoded blocks are placed on its set of n remote peers.

At each cycle, backup and restore requests are generated among peers according to the previously defined Poisson processes. Note that a backup request is uniformly distributed among the set of online peers, while a restore request is uniformly distributed among the set of peers whose data has already been fully backed-up (*i.e.* all blocks have been offloaded on remote peers). The rate of backup requests is set so that each peer requests to store about three archives of 1GB per month on average (*i.e.* $\lambda_{\text{backup}} = 0.4$ and $\lambda_{\text{restore}} = 0.05$ for the Skype trace).

Once all backup and restore requests have been distributed among the peers, each device (*i.e.* peer or gateway) transfers as much data as its upload bandwidth allows for the duration of the cycle (*i.e.* since residential connections are asymmetric, we assume that the system is bounded by the upload bandwidth of devices). If the remote device is offline, the client device has to wait until its reconnection.

Our simulations were conducted with a cycle time-step set to 5 minutes. Figures report the average behavior over 50 simulations. We evaluate the three systems according to the following four metrics:

Time to backup, noted TTB. A backup request is considered successful when the data is stored safely. Safe storage is achieved when n blocks have been uploaded to the CDN and the remote peers for *CDNA*, to the remote gateways for *GWA*, and to the remote peers for the *P2P* thus satisfying the targeted availability described earlier. The time to backup is evaluated as the difference between the time the n^{th} block has been downloaded and the time of the backup request.



(a) TTB (Skype trace)
(b) TTR (Skype trace)
(c) TTB (Jabber trace)
(d) TTR (Jabber trace)

Figure 5. CDF of Time to Restore and Time to Backup for two traces of Instant Messaging.

Time to offload, shown on the same plots as TTB. A variant measure for both *CDNA* and *GWA* is the time to fully offload an archive to the remote peers. This means that no data is left on the CDN or on gateways, accordingly. We note this variant $CDNA_p$ and GWA_p respectively.

Time to restore, noted TTR. A restore request is considered successful as soon as the data is restored safely at the user's place, that is to say when at least k blocks have been retrieved on the client peer, or on the gateway of the client peer for *GWA*. The time to restore is evaluated as the difference between the time the k^{th} block has been downloaded and the time of the restore request. We measure this time for random files after a long enough period, so that the selected files have been offloaded to peers. This represents the worst case for TTR, and this assumption reflects the fact that restore requests are more likely to happen long after backups.

Data buffered. It describes the size of the buffer that is required on gateways, for dimensioning purposes. We measured the maximum and the average amount of data stored on those buffers.

V. Results

A. Time to backup & restore results

We first evaluate the performance of our approach to backup and restore data *w.r.t.* time. We plot the experimental cumulative distribution function (CDF) for TTB and TTR arguably the most critical metrics from the users' standpoint. CDF(t) represents the cumulative number of requests that have been completed after t hours.

Figure 5a depicts the TTB for the Skype trace. Results show that our gateway assisted approach improves the TTB over the *CDNA* approach[3]. Both considerably improve the performance over the *P2P* approach. Our *GWA* completes 90% of backup in 30 hours, the *CDNA* completes 90% of backups in 60 hours and the *P2P* completes 90% of backups in 140 hours. A consequence of this improvement is that data is backed up faster, reducing the window of potential losses.

Along the performances of *P2P*, *CDNA* and *GWA*, we also indicate those of $CDNA_p$ and GWA_p. In *GWA* and *CDNA*, once the data is backed up (*i.e.* it is stored safely at some place being the remote gateways or the CDN), the backup process continues by offloading the data hosted on the remote gateways or on the CDN to the remote peers (*i.e.* until no data is left on the remote gateways or on the CDN). This process takes some time and its total duration is shown as GWA_p and $CDNA_p$. It is interesting to observe that while the *CDNA* does not enhance this time when compared to *P2P* (*P2P* curves and $CDNA_p$ curves are superimposed on all plots), our *GWA* approach improves this time significantly, reducing it from 140 hours to 90 hours (for 90% of backups to be offloaded).

We plot the time to restore for the Skype trace on Figure 5b. Results show that the TTR of our *GWA* approach is significantly reduced when compared to *GWA* and *P2P* approaches. Our *GWA* allows 90% of restores to be completed in less than 3 hours while both *CDNA* and *P2P* require 40 hours to complete 90% of restores. This makes the system much more user-friendly, when it comes to retrieving lost files.

In order to show that these results are not trace-dependent, we run the same set of experiments on a trace exhibiting a

[3]Note that results for *CDNA* are consistent with simulations made in [4], with an improvement factor between 2 and 3 over *P2P* storage

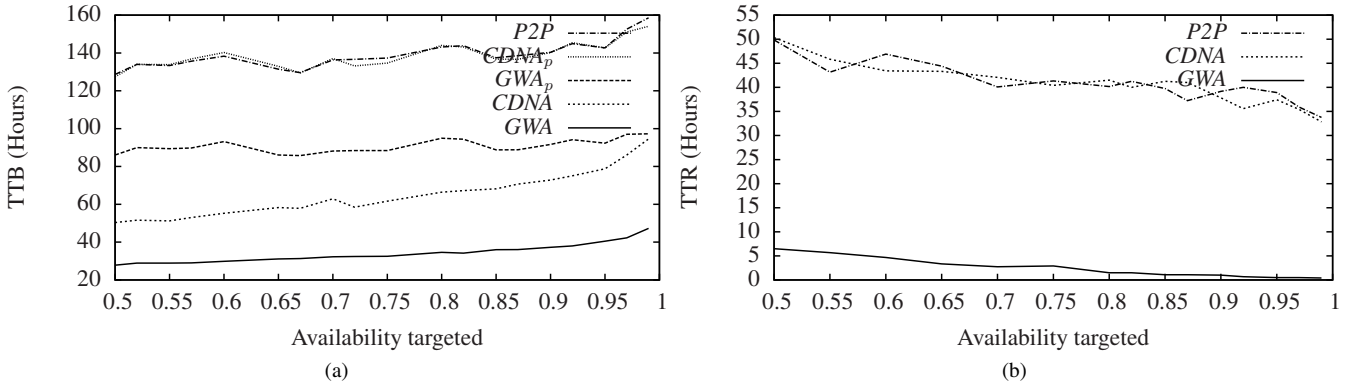


Figure 6. Results (Skype trace), as a function of availability target, 90th percentile. Labels follow the order of the curves.

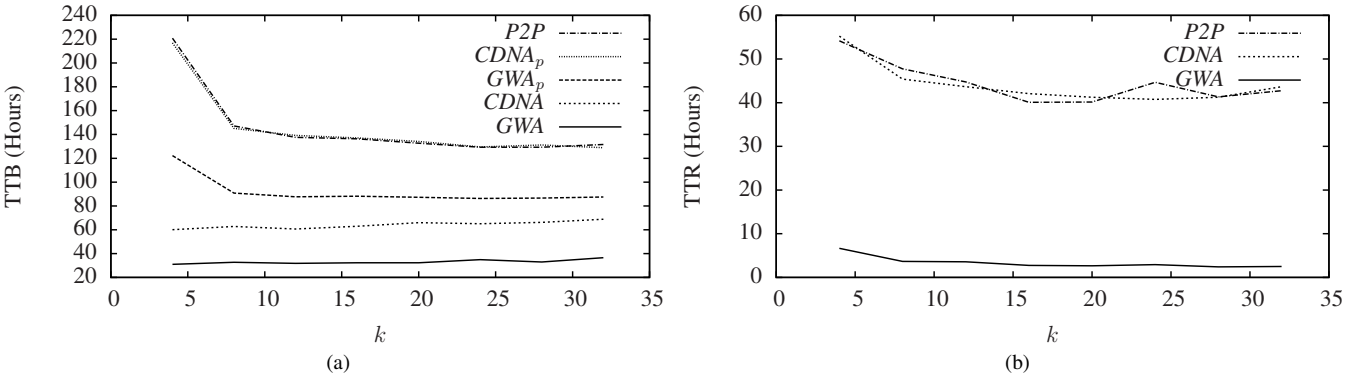


Figure 7. Results (Skype trace), as a function of k, 90th percentile.

lower peer availability, namely the Jabber trace. We observe similar results. Previous observations hold as the time to backup (Figure 5c) for our *GWA* is reduced when compared to both *CDNA* and *P2P*. Similarly, the time to restore is also reduced (Figure 5d). Overall, the absolute times are increased when compared to results observed over the Skype trace because of the lower average availability of peers in this trace (25% instead of 50%).

Note that the absence of full convergence in some cases is due to some peers leaving the network for good before the backup is complete and because the trace duration is limited.

In the remaining part of this section, we evaluate the impact of other parameters, namely the targeted availability A_t, the number of blocks k, the number of peers involved in the system and the amount of data to backup and restore (*i.e.* the archive sizes) on both the TTB and TTR. We vary parameters one by one, keeping the other parameters to the values previously defined. Previous results have shown that some small part of backups may take significantly more time to complete than average; this somehow introduces a bias in observation of the system. In the following results, we will then consider the 90^{th} percentile for all approaches, in order to gain clear insights from systems' trends (*i.e.* the times displayed are the time so that 90% of the operations have been completed). The behavior, for one set of parameters, of the remaining 10% can be seen on the previously described CDF.

B. Influence of targeted availability

On Figure 6a, we plot the opposed variations of TTB and TTR when the targeted availability changes. We observe that the TTB increases when the targeted availability increases. This is due to the fact that achieving a higher availability for the file requires uploading more data, to introduce more redundancy. However, as shown on Figure 6b, increasing the targeted availability (*i.e.*, the amount of redundancy introduced) helps to reduce the TTR. As a result the targeted availability is a tradeoff between having a low TTR and keeping the TTB reasonable. We observe that the improvement of our *GWA* approach is very significant over *CDNA* or *P2P* approaches. This is due to the fact that our approach represents a paradigm shift: the data is now considered as restored when stored in the home of the requester (on its gateway). Indeed, this prevents exterior factors such as block losses in the system from having any impact. When the requesting peer comes back online, it can seamlessly fetch data in order of minutes from its gateway, making this step nearly transparent when compared to operations at WAN speed.

Another aspect is impacted by the availability setting of the storage system: increasing the targeted availability requires to store more redundancy blocks thus increasing the storage costs

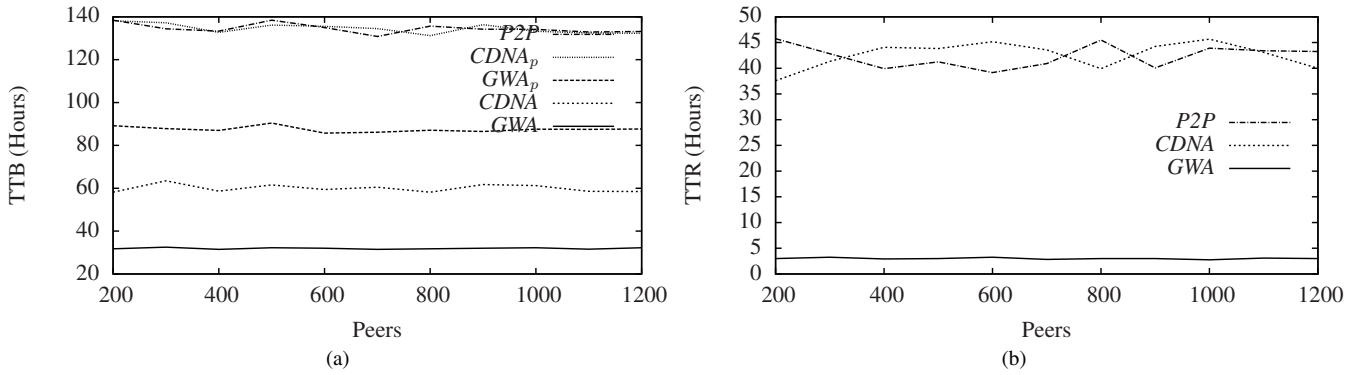


Figure 8. Results (Skype trace), with a varying network size, 90th percentile

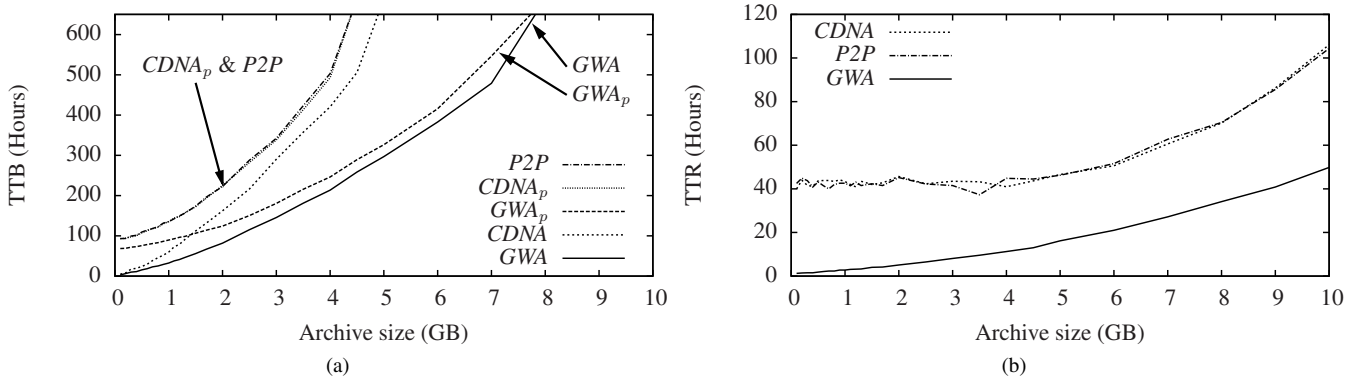


Figure 9. Results (Skype trace), with a varying amount of data to backup, 90th percentile

per peer, as shown on Table I. Hence, keeping the targeted availability reasonably low helps to keep storage costs per peer low thus increasing the total amount of data that can be backed up. In the remaining simulations, we choose to use a medium value of 0.7 for that targeted availability.

Table I
ADDITIONAL STORAGE COSTS INDUCED BY VARIOUS TARGET AVAILABILITIES, WHEN COMPARED TO OUR REFERENCE TARGET OF 0.7

Target availability	0.5	0.7	0.9	0.95	0.99	0.995
Jabber (k=8)	−12%	-	27%	39%	67%	79%
Skype (k=16)	−11%	-	14%	20%	34%	40%

C. Influence of code length k

Increasing the code length k (*i.e.* the number of blocks required to decode) slightly improves both TTB and TTR, as shown on Figures 7a and 7b. However, increasing k also increases some side costs such as coding and decoding costs. As a consequence, we choose to set $k = 16$ to leverage the code properties while retaining low coding and decoding costs.

D. Influence of the size of the network

Next, we evaluate the influence of the number of peers participating in the system by varying the number of peers from 200 to 1269 (all the peers of our trace). We also set the parameters for the restore and backup events so that on average each peer performs around 3 backup requests per month and 3 restore requests per year, similarly to what was done for all other simulations. Our results are shown on Figures 8a and 8b. Both TTB and TTR remain constant. Our simulations are limited by the total number of peers present in the trace; yet, we observe on this moderate scale that our system is likely to be scalable in the number of participating peers, as no bottleneck occurs at some gateways, which could have increased the TTR. Note that these constant TTB and TTR are consistent with the fact that the load (backup and restore requests) per peer is constant.

E. Influence of the amounts of data to backup and restore

In order to assess the gains with respect to the usability of *GWA*, we plot the evolution of the TTB (Figure 9a) and the TTR (Figure 9b) as the amount of data to backup increases. Our architecture enables to leverage the difference of bandwidth between the local network and the Internet connection in order to use the uplink more efficiently (21h/24h instead of only 12h/24h); clearly, in this case, it enables users to backup larger amounts of data (here by a factor of 2). Note that our simulations are limited by the trace length (28 days).

F. Resulting needs for storage dimensioning on buffers

For dimensioning purposes, we evaluate how much disk space should be provisioned on the gateways in order to implement our backup service. Figure 10 shows the CDF of maximal

storage at gateways resulting from 1GB archive backups, at any time, on the Skype trace (*i.e.* the worst case space occupied on any gateway at any simulation step). Storage needs remain within reasonable bounds: 99% of gateways have stored at most 2.5GB of data at anytime. In a deployment scenario, if we limit the provisioned storage to 2.5GB at each gateway, the remaining 1% of gateways could ask peers pushing blocks to delay their query until their buffers have emptied, without impacting performance since it would impact less than 1% of peers and would occur only exceptionally.

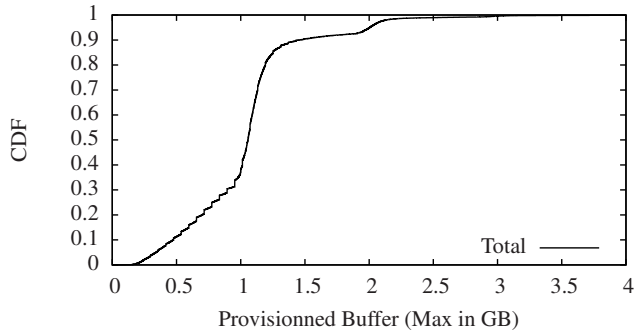


Figure 10. Maximal storage measured on gateways at any time (Skype trace)

Buffers are used in burst mode for relatively short periods of time, as shown on Figure 11, which plots the average of the same storage functionalities for the whole trace period: the average is negligible compared to the maximum presented on the previous figure. Furthermore, we observe that data is effectively offloaded since nothing remains on buffers when no new backups are requested (on the simulations, we performed backups only for the 13 first days) thus confirming results shown on previous curves for GWA_p.

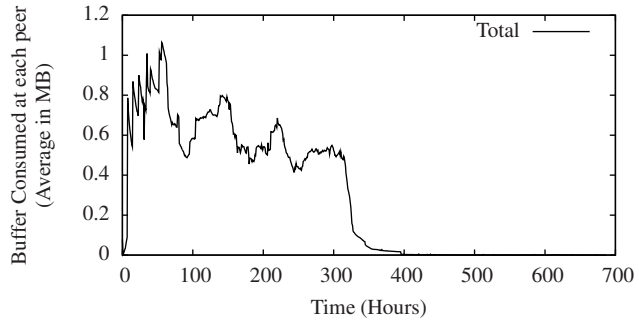


Figure 11. Average storage measured on gateways (Skype trace)

These results are related to our model where each gateway hosts one peer. The storage requirements would increase if multiple peers are behind each gateway. However, we study residential gateways, which are shared among a very small number of users. The storage requirements increase at most linearly with the number of users and hence would remain rather low as only a few peers are behind each gateway.

VI. Discussion

Our results clearly indicate that our proposal efficiently distributes the centralized buffer scheme of [4], while increasing general backup performances and represents a significant improvement over previous approaches.

The Web architecture, in particular when considering CDN, relies on cache servers close to clients [23]. However, these servers are located within the Internet and cannot benefit from the difference of throughput between the local home network and the Internet: our system relies on the specific place of the gateway in between the home network and the Internet to leverage this difference. Moreover, cache servers are generally passive (*e.g.*, HTTP proxy) while in our system, the gateway is not only a cache but also an active participant that can serve directly other peers when data is requested.

Additionally, from the user's standpoint, our storage system could enable the bandwidth usage to be smoothed to provide users with a more transparent service (*i.e.* using the upload for backup when users are not using their computer/Internet connection). Indeed, using an important part of the upload bandwidth to quickly complete the backup operation, may severely affect the user's experience of Internet browsing or activity. A user's gateway is able to upload, as long as there is some available bandwidth and even if the user's computer is turned off (typically nightly). A similar advantage, appealing for Internet and service providers [24], is that such an architecture enables the implementation of scheduling policies for delaying transfers from gateways to the Internet so as to smooth the usage of the core network.

Lastly, this method also solves another issue that might appear when the distributed application operates worldwide. It has been shown that peers' availability patterns can vary according to local time (depending on geographical location) [17], [18]; in systems where some resources are insufficiently replicated, this could lead to an asynchrony of presence between a requesting peer and another peer hosting the resource. At best, the overlap of presence of both peers is sufficient to download the file, while it may also require a few sessions to complete, due to insufficient time overlap. As our *GWA* proposal relies on the hosting peer to upload the requested file to a more stable component (its gateway), asynchrony is no longer an issue as gateways provide stable rendezvous point between requesters and providers. This is of interest for delay tolerant applications such as backup, allowing the service to be operated with much lower costs on storage. Beyond the practical problem of using gateways in home environments, our solution then makes the case for leveraging clouds of stable components inserted in the network, to make them act as buffers in order to mask availability issues introduced in dynamic systems.

VII. Related Work

We compared our approach to the peer-assisted one presented in [4] that leverages a central server and offloads backed up data to peers when they become available. Such a server-centric approach is also to be found in [14], where authors

propose an hybrid architecture coupling low I/O bandwidth but high durability storage units (being an Automated Tape Library or a storage utility such as Amazon S3 [25]) with a P2P storage system offering high I/O throughput by aggregating the I/O channels of the participating nodes but low durability due to the device churn. This study provides a dimensional and system provisioning analysis via an analytical model that captures the main system characteristics and a simulator for detailed performance predictions. The simulator does not use real traces but synthetic ones, mainly to be able to increase the failure density and to reveal the key system trends. This work explores the trade-offs of this hybrid approach arguing it is providing real benefits compared to pure P2P systems [6]–[10]. Durability of the low I/O bandwidth unit is considered as perfect, but it always comes at a certain cost. In our approach, we do not assume we have such nodes and show that our approach is sustainable under real availability traces. Finally, FS2You [15] proposes a BitTorrent-like file hosting, aiming to mitigate server bandwidth costs; this protocol is not designed to provide persistent data storage.

The increasing power of residential gateways has enabled numerous applications to be deployed on them. This may allow savings in terms of power. One widely deployed system is the implementation of BitTorrent clients in those boxes (see *e.g.* FON [26]), which avoids the user to let her computer powered on [16]. Another example is the concept of Nano Data Centers [5], where gateways are used to form a P2P system to offload data centers. Similarly, some approaches were proposed to move tasks from computers to static devices as *set-top boxes*, for VoD [27] and IPTV [28]. Yet, those applications fully run on gateways while, in our approach, the gateway only acts as buffering stage.

VIII. Conclusion

This paper addresses the problem of efficient data backup on commodity hardware. It has been widely acknowledged that availability of transient peers is a key parameter, that can by itself forbid a realistic service deployment if too low. We propose to address this inherently transient behavior of end peers by masking it through the use of more stable hardware, already present in home environments, namely gateways. Our experiments, based on real availability traces, show that this architectural paradigm shift, significantly improves the user experience of backup systems over previous approaches, while remaining scalable.

We also provide a new trace of gateway availabilities, which is of interest for trace-based simulation of systems built upon gateways. As future works, it would be interesting to acquire a trace of user behaviors with respect to storage demand in peer-to-peer systems.

References

[1] W. J. Bolosky, J. R. Douceur, D. Ely, , and M. Theimer, "Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs," in *SIGMETRICS*, 2000.

[2] H. Huang, W. Hung, and K. G. Shin, "FS2: dynamic data replication in free disk space for improving disk performance and energy consumption," in *SOSP*, 2005.

[3] L. Pamies-Juarez, P. García-López, and M. Sánchez-Artigas, "Availability and Redundancy in Harmony: Measuring Retrieval Times in P2P Storage Systems," in *P2P*, 2010.

[4] L. Toka, M. Dell'Amico, and P. Michiardi, "Online Data Backup: A Peer-Assisted Approach," in *P2P*, 2010.

[5] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the Internet with Nano Data Centers," in *CoNext*, 2009.

[6] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: an architecture for global-scale persistent storage," *SIGPLAN Not.*, vol. 35, pp. 190–201, 2000.

[7] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *SOSP*, 2001.

[8] L. P. Cox, C. D. Murray, and B. D. Noble, "Pastiche: making backup cheap and easy," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 285–298, 2002.

[9] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A cooperative internet backup scheme," in *Usenix ATC*, 2003.

[10] M. Landers, H. Zhang, and K.-L. Tan, "PeerStore: Better Performance by Relaxing in Peer-to-Peer Backup," in *P2P*, 2004.

[11] C. Blake and R. Rodrigues, "High availability, scalable storage, dynamic peer networks: pick two," in *HOTOS*, 2003.

[12] K. Tati and G. M. Voelker, "On Object Maintenance in Peer-to-Peer Systems," in *IPTPS*, 2006.

[13] P. Maille and L. Toka, "Managing a Peer-to-Peer Data Storage System in a Selfish Society," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 7, pp. 1295 –1301, 2008.

[14] A. Gharaibeh and M. Ripeanu, "Exploring data reliability tradeoffs in replicated storage systems," in *HPDC*, 2009.

[15] F. Liu, Y. Sun, B. Li, B. Li, and X. Zhang, "FS2You: Peer-Assisted Semipersistent Online Hosting at a Large Scale," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, pp. 1442–1457, 2010.

[16] G. Fedak, J.-P. Gelas, T. Herault, V. Iniesta, D. Kondo, L. Lefevre, P. Malécot, L. Nussbaum, A. Rezmerita, and O. Richard, "DSL-Lab: A Low-Power Lightweight Platform to Experiment on Domestic Broadband Internet," in *ISPDC*, 2010.

[17] J. R. Douceur, "Is remote host availability governed by a universal law?" in *SIGMETRICS*, 2003.

[18] R. Bhagwan, S. Savage, and G. Voelker, "Understanding Availability," in *IPTPS*, 2003.

[19] M. Dischinger, A. Haeberlen, K. P. Gummadi, , and S. Saroiu., "Characterizing Residential Broadband Networks," in *IMC*, 2007.

[20] S. Guha, N. Daswani, and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *IPTPS*, 2006.

[21] W. K. Lin, D. M. Chiu, and Y. B. Lee, "Erasure Code Replication Revisited," in *P2P*, 2004.

[22] H. Weatherspoon and J. Kubiatowicz, "Erasure Coding Vs. Replication: A Quantitative Comparison," in *IPTPS*, 2002.

[23] T. Leighton, "Improving Performance on the Internet," *CACM*, vol. 52, 2009.

[24] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram, "Delay Tolerant Bulk Data Transfers on the Internet," in *SIGMETRICS*, 2009.

[25] "Amazon Web Services." http://s3.amazonaws.com.

[26] "FON," http://corp.fon.com.

[27] V. Janardhan and H. Schulzrinne, "Peer assisted VoD for set-top box based IP network," in *P2P-TV*, 2007.

[28] M. Cha, P. Rodriguez, S. Moon, and J. Crowcroft, "On next-generation telco-managed P2P TV architectures," in *IPTPS*, 2008.