Performance evaluation of a peer-to-peer backup system using buffering at the edge

Anne-Marie Kermarrec, Erwan Le Merrer, Nicolas Le Scouarnec, Romaric Ludinard, Patrick Maillé, Gilles Straub, Alexandre van Kempen

Performance evaluation of a peer-to-peer backup system using buffering at the edge

Anne-Marie Kermarrec[b], Erwan Le Merrer[a], Nicolas Le Scouarnec[a], Romaric Ludinard[b], Patrick Maillé[c], Gilles Straub[a], Alexandre Van Kempen[a]

[a] *Technicolor*
[b] *INRIA Rennes*
[c] *Institut Telecom, Telecom Bretagne*

Abstract

The availability of end devices of peer-to-peer storage and backup systems has been shown to be 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. We model such a system, for quick dimensioning and estimation of gains. We then evaluate our proposal using statistical distributions based on real world traces, as well as a trace of residential gateways for availability (that we have collected and now make available). Results show that the time required to backup data in the network is substantially improved, as it drops from 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.

Keywords: Storage, Backup, Availability, Peer-to-peer.

1. 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 time. While cloud backup systems are increasingly adopted by users (*e.g.,* justcloud, SugarSync, Egnyte HybridCloud, 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]. A particularly illustrative example is the Wuala case: the Wuala company gained fame by proposing a peer-assisted (advertised as fully peer-to-peer) and practical storage service; nevertheless, this technical choice was abandoned to move to a centralized architecture [4], probably for cost/performance matters. Beside this initial example and academic systems, we are not aware of a peer-to-peer storage system deployed at large scale for common needs.

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 [5]. 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 performance problem and investigate a new way of performing efficient backup on commodity hardware in a fully peer-to-peer way. Other specific issues with peer-to-peer solutions include security or QoS [6], but are out of the scope of this article.

In this article, we propose a new architecture for peer-to-peer backup, where residential gateways are turned into a *stable buffering layer* 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 [7]. Our idea is to temporarily store data on gateways to compensate for peers transient availability. In this article, 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. Such a system enhances the backup system's performance along two lines:

- The network connection can be used more efficiently: if devices upload data constinuously while they are up, the available bandwidth can be exploited typically 21h/day instead of only 6 to 12h/day on average, based on actual measured availabilities. This leads to significant enhancements. For example, we observe that the time to backup a 1GB archive is reduced from few weeks 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.

Our proposal differs from existing approaches [4, 5, 8, 9, 10, 11, 12, 13] 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. 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 article is structured as follows. In Section 2, we briefly review some pieces of work that motivated our proposal. In Section 3, we detail our architecture and sketch the storage system. We then propose in Section 4 a model for this system, in order to estimate its performance. Section 5 introduces a framework for comparison of our proposal to that of competitors, and presents our evaluation study. We discuss respectively some specific points and related work in Section 6 and Section 7. Finally, we conclude the article in Section 8.

2. 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 [8, 9]. However, it has been acknowledged since then [14, 15] 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 [16]; this nevertheless excludes many participants and significant amounts of exploitable resources [1, 2].

Hybrid architectures, where both servers and peers coexist, have been proposed in various contexts, in order to move towards practical system deployment while still leveraging users' resources [17]. The problem of sharing files while mitigating the load of central servers is addressed in [18]. This article 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 [5]. In this 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.

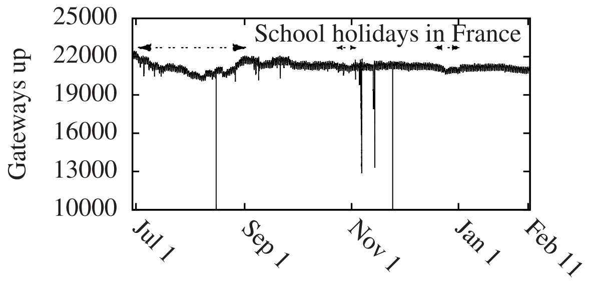
Finally, another aspect of interest is the network setting of home networks. Residential gateways connect home local area networks (LAN) to the Internet. They act as routers between their WAN interface (Cable or DSL) and their LAN interfaces (Ethernet and WiFi). They started to be deployed in homes to 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 [7, 19, 20, 21].

3. A gateway assisted system

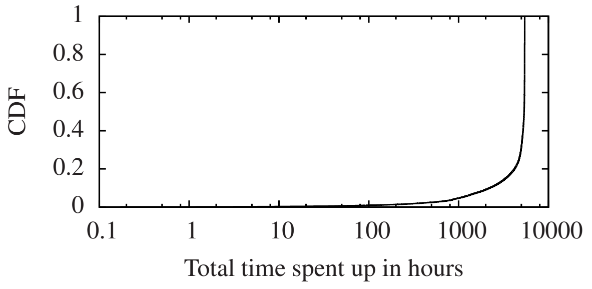
3.1. Stability of residential gateways

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.

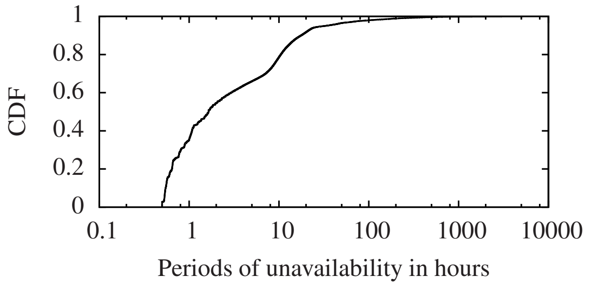
[1]This trace and additional information can be found at the following URL `http://www.thlab.net/~lemerrere/trace_gateways/`.



(a) Number of gateways simultaneously connected, across the monitored period (7.5 months).



(b) CDF of uptime periods in the trace.



(c) CDF of dowtime periods in the trace.

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.

We obtained the uptime patterns of $25,000$ gateways for 7.5 months, covering week-patterns [22, 23], 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, as seen on Figure 1b and 1c (a gateway that has rebooted participates with its corresponding number of uptimes periods to those plots), 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 [7], where the authors used traces from a biased sample (only BitTorrent users) [24]. This has to be contrasted with the low to medium availabilities of peers generally recorded in the literature, as *e.g.,* 27% in [5], or 50% in [25].

For completeness, we provide statistics on this trace by using the Failure Trace Archive toolbox; this is useful to compare to the other 9 availability traces which are available and studied in [26].

Mean	Tr. Mean	Median	Std	Coeff. Var.	Int. Quart.	Max	Min	3rd Moment
154.64	91.01	10.62	371.72	2.40	80.20	3522.41	0.14	3.80

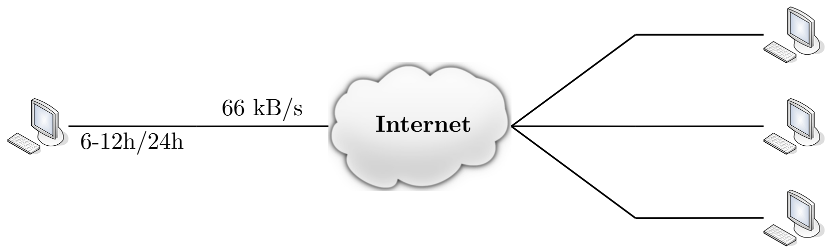
Figure 2: Availability statistics extracted from the raw trace of gateways. (Values are given in hours.)

For modeling uptime periods in this trace, two distributions are good fits:

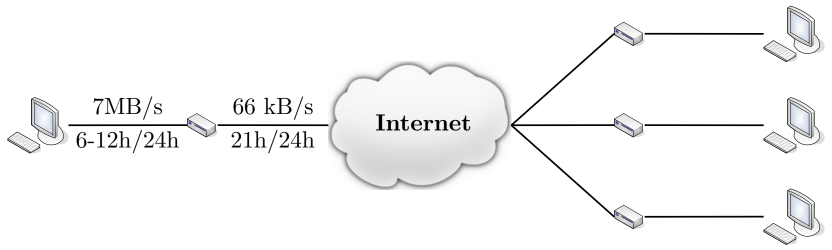
- Weibull (p-value: KS-test=0.22, AD-test=0.34 [26]), with parameters $scale = 46.96$ and $shape = 0.42$.
- Log-Normal (p-value: KS-test=0.31, AD-test=0.48) with parameters $mean = 2.65$ and $std.dev. = 2.29$.

3.2. System rationale

In this article, we propose to decentralize the buffer logic implemented in [5] by a CDN, in order to provide a reliable backup system despite the dynamic nature of peers composing 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 3a) 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 3b). 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.



(a) With passive gateways



(b) With active gateways

Figure 3: 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).

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 which 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, which is consistent with the 2Mb/s and 3.5Mb/s values respectivelty **advertised** by providers, in OECD studies [27]). Exact throughput numbers are bound to evolve positively, but the crucial factor is the steady gap between LAN and WAN speeds. Our architecture exploits the major difference of throughput between the LAN and the Internet connection (WAN)[2] by offloading tasks from the peer to the corresponding gateway quickly, thus using the Internet

[2]Note that even if fiber technology can solve part of the asymmetry problem, it is still far from being the norm in most countries (please refer to OECD studies for numbers [27])

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 modify their habits (*e.g.,* users must leave their computers powered on during the whole night to be able to upload large archives, even if they usually turn them off); 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. 3a); 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. 3b). 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 (noted *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.

3.3. 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 [28] 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 article, we set a target A_{target} for the system-wide availability so that n must be the smallest n ensuring that $A > A_{target}$. 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 [14], so a reasonable trade-off should be considered [3].

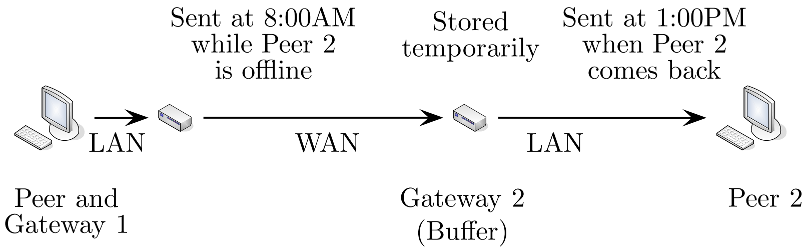

Sent at 8:00AM while Peer 2 is offline
Stored temporarily
Sent at 1:00PM when Peer 2 comes back
LAN
WAN
LAN
Peer and Gateway 1
Gateway 2 (Buffer)
Peer 2

Figure 4: 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 4). 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 4) 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 system 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.** 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.

4. Markov modeling of the storage system

We now present a mathematical model for the gateway-assisted storage system just introduced. Such a model is particularly interesting to *(i)* estimate order of magnitudes for time to backup and time to restore, based on system's core parameters, and to *(ii)* dimension gateway hard drives, in order for the system to be viable.

4.1. Model parameters

In our model, we consider a very large number N of peers, such that the contribution of each individual peer on the total backup requests is infinitesimal. That reasonable assumption will lead to some simplifications of the model.

The availability behavior of the peers is modeled by a simple random process where each peer remains connected during a random time, that we assume follows an exponential distribution with parameter λ, independent of all the rest. Similarly, the durations of offline periods are modeled by independent exponentially distributed random variables with parameter μ. As a result, the average availability of each peer is $\bar{p} = \frac{\mu}{\lambda+\mu}$.

The backup operation requests are supposed to occur over time independently among peers, and the average number of backup requests per peer and per time unit is denoted by θ_b. Note that the model allows heterogeneous backup behaviors among peers, θ_b being an average over all peers. However, we impose that this average number of backup requests remain stationary over time (*i.e.,* no "peak backup hours"), which is a reasonable assumption if we consider that peers are spread worldwide.

Each backup request is supposed to consist in one archive of total size S, that is split into k blocks of individual size s. Redundancy is then added to form $n > k$ blocks of size b, and each block is sent to the gateway of a peer, selected randomly and uniformly among all peers[3].

Take the point of view of a particular peer, and consider a time period of duration $t > 0$ during which R backup requests are issued in the whole system. Define $r := R/N$. On that period, the number of blocks sent to the considered peer's gateway for storage then follows a binomial distribution $B(nrN, 1/N)$, that converges to a Poisson distribution with mean nr when N gets large. But notice then $r = R/N$ tends to $t\theta_b$, so that the number of blocks received by the gateway on any period of duration t follows a Poisson distribution with mean $nt\theta_b$. Formulated differently, the time between two block receptions follows an exponential distribution with parameter $n\theta_b$.

Since LAN speeds are considerably larger than WAN transmission rates, the former are considered infinite (*i.e.,* the transfer times between a peer and its gateway are neglected). All gateways are assumed to benefit from the same upload transfer rate, that we denote by d_u. The download rates of gateways is considered not to be a limiting factor (the bottleneck therefore is at the sending gateway). Finally, we assume that the n blocks to store onto the system are sent in parallel to the recipients' gateways, the transfer therefore taking a constant time $\frac{ns}{d_u}$.

Finally, one dimensioning parameter will be the memory size to put into the gateway, in order to keep the backup blocks received from the other peers. We focus on the download memory here, whose capacity, in number of blocks, is denoted by C. Some upload memory would also be needed in case the peer sends a backup request and goes offline before the packets are sent to the storing peer gateways. We can reasonably dimension that

[3]Since the number of peers is assumed very large, the probability that two blocks are sent to the same peer, or that a peer has to store one block it emitted, is negligible.

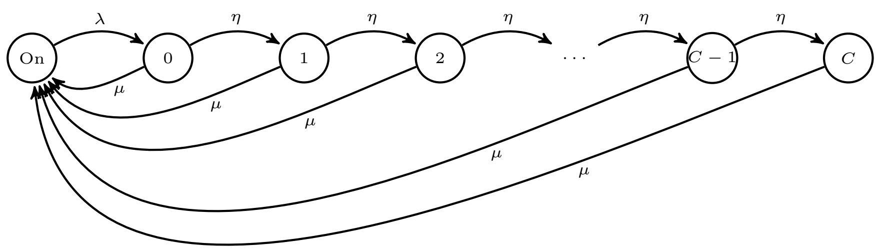


Figure 5: Transition diagram for the Markov chain describing the evolution of $K(t)$, the number of blocks received and buffered in the peer's gateway. That number is 0 as soon as the peer is online (state "On").

uplink memory to n blocks, which is sufficient if there is only one ongoing backup request at any time: this occurs for example if new backups replace the previous ones (even if not completed).

In case the same memory can be indifferently used for uplink and downlink, a sufficient dimensioning is to set the total memory to $\max(C, n)$ blocks. Indeed:

- under our assumptions the downlink buffer of the gateway is empty as soon as the peer is online;
- the worst case is then when the peer goes offline just after sending a backup request, and its gateway receives several blocks. But even then, the memory empties at speed d_u which, in the very unlikely case when more than n blocks are then simultaneously received from other nodes, would just slightly slow down those transfers.

4.2. Markov modeling

Let us consider a given peer. We analyze the evolution over time of the number of blocks kept at the gateway of that peer, waiting for that peer to appear online. When the peer is offline:

- either it comes back online, after some random time that follows an exponential distribution with parameter μ and then the gateway memory is emptied;
- or a new block is received by the peer's gateway, after some random time exponentially distributed with parameter $\eta := n\theta_b$, as developed before.

If the download buffer of the gateway can store C blocks, the number $K(t)$ of blocks in that buffer at time t is a continuous-time Markov chain, whose transition diagram is drawn in Figure 5.

As being irreducible and with a finite number of states, the Markov chain is ergodic and therefore admits a steady-state distribution. We denote by p_i the steady-state probability of state i. In addition, we notice that the probability p_{On} that the peer is online, simply equals $p_{\text{On}} = \frac{\mu}{\lambda+\mu}$, due to our assumptions on the peer's behavior.

Using the stationary conditions, we easily obtain that for all $0 \leq i \leq C-1$,

$$p_i = \left(\frac{\eta}{\eta+\mu}\right)^i \frac{\lambda}{\eta+\mu} \frac{\mu}{\lambda+\mu}$$

and

$$p_C = \frac{\eta}{\mu} p_{C-1} = \left(\frac{\eta}{\eta+\mu}\right)^C \frac{\lambda}{\lambda+\mu} \tag{2}$$

4.2.1. Dimensioning the gateway reception buffers

Because of the PASTA (Poisson Arrivals See Time Averages) property, the probability p_C expressed in (2) is also the probability that a block is lost at the peer. Dimensioning the downlink buffer of the gateway is then straightforward: with a target loss rate of $\varepsilon_{\text{target}}$, we find:

$$p_C \leq \varepsilon_{\text{target}} \qquad \Leftrightarrow \qquad C \geq \frac{\log(\varepsilon_{\text{target}}(1+\mu/\lambda))}{\log(\eta/(\eta+\mu))}.$$

The values of C depending on the block arrival rate η are plotted in Figure 6, for different target loss values, and with average online and offline durations of 17 and 7 hours, respectively, leading to the availability $p_{\text{On}} \approx 0.7$.

4.2.2. Backup duration

We focus here on the time needed before the backup data are safely stored, *i.e.,* the time between a backup request and the instant when all (or a sufficient part of) the data is stored on the receiving peers.

The duration of the transmission to the gateways of the receiving peers is constant under our assumptions, and equals $\frac{ns}{d_u}$. After that time, the gateways need to offload the received blocks to the peer, which occurs with negligible time when the peer is online. Two cases are possible upon the arrival of a block (assuming the block is not rejected, which is reasonable if we took $\varepsilon_{\text{target}}$ small enough):

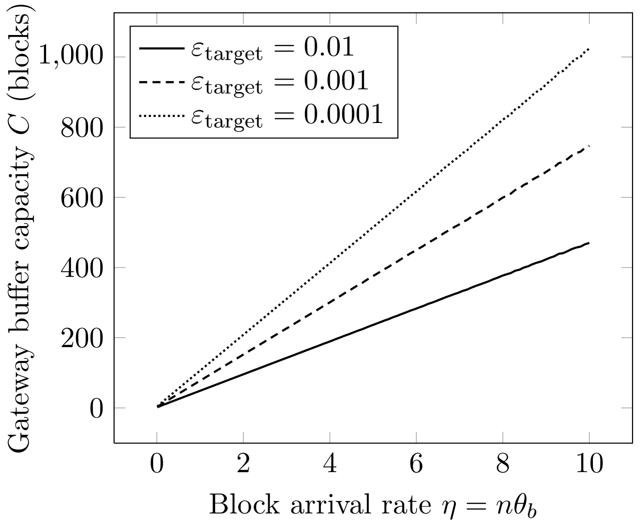


Figure 6: Buffer dimensioning ($\mu = 1/12$ hours^{-1}, $\lambda = 1/12$ hours^{-1}).

- if the peer is online when the block is received –which is the case with probability $\frac{\mu}{\lambda+\mu}$–, then there is no extra delay incurred;
- on the contrary, if the peer is offline upon its gateway receiving a block, then the time T before that block is stored by the peer is a random variable, exponentially distributed with parameter μ, *i.e.,* $\mathbb{P}(T \leq t) = 1 - e^{-\mu t}$.

Consequently, through a conditioning on the number of peers that are online when their gateway receives a block of the considered backup request, we can express the random part T_{backup} of the backup duration (*i.e.,* removing the constant time $\frac{ns}{d_u}$ to send the blocks to the receiving gateways). If we consider that the backup is over when the n peers received their assigned block, we have:

$$\mathbb{P}(T_{\text{backup}} \leq t) = \sum_{i=0}^{n} \underbrace{\binom{n}{i} \left(\frac{\lambda}{\lambda+\mu}\right)^{i} \left(\frac{\mu}{\lambda+\mu}\right)^{n-i}}_{\mathbb{P}(i \text{ peers among } n \text{ are offline})} \underbrace{\left(1 - e^{-\mu t}\right)^{i}}_{\mathbb{P}(\text{they come online before } t)},$$

with $\binom{n}{i} = \frac{n!}{i!(n-i!)}$.

That distribution is plotted on Figure 7 for different values of n.

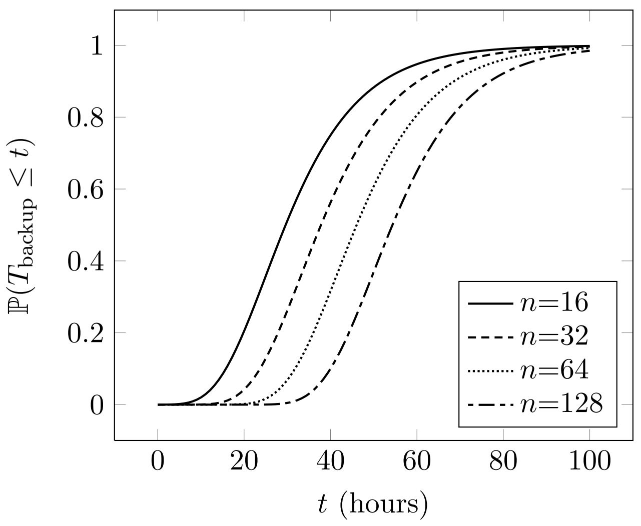


Figure 7: Distribution of the backup duration (time before all n blocks are stored on peer's machines), to be added to the constant component $\frac{ns}{d_u}$. (Parameters: $\lambda = 1/12$ hours^{-1}, $\mu = 1/12$ hours^{-1}.)

Note that we could also compute the time before $\ell \leq n$ blocks are stored by the peers: taking $\ell \geq k$, we could consider that enough data is safely stored. The distribution of that time $T_{\text{backup}}(\ell)$ is then such that:

$$\mathbb{P}(T_{\text{backup}}(\ell) > t) = \sum_{i=n-\ell+1}^{n} \underbrace{\binom{n}{i} \left(\frac{\lambda}{\lambda+\mu}\right)^{i} \left(\frac{\mu}{\lambda+\mu}\right)^{n-i}}_{\mathbb{P}(i \text{ peers among } n \text{ are offline})} \Phi^{i}_{n-\ell+1}(t), \quad (3)$$

with $\Phi^i_j(t)$ the probability that at least j peers among i initially offline remain offline during t, *i.e.*:

$$\Phi^i_j(t) = \sum_{m=j}^{i} \binom{i}{m} e^{-m\mu t}(1 - e^{-\mu t})^{i-m}. \quad (4)$$

The index i in (3) represents the number of peers that are offline after the gateways receive the blocks (*i.e.*, some time $\frac{ns}{d_u}$ after the backup request).

For example, the distribution of the time before a proportion 15/16 of the blocks are stored on peers' disks (*i.e.*, $\ell = \frac{15n}{16}$, which corresponds to a sufficient backup with a redundancy factor of 16/15) is plotted in Figure 8, illustrating (in comparison with Figure 7) how redundancy helps reduce

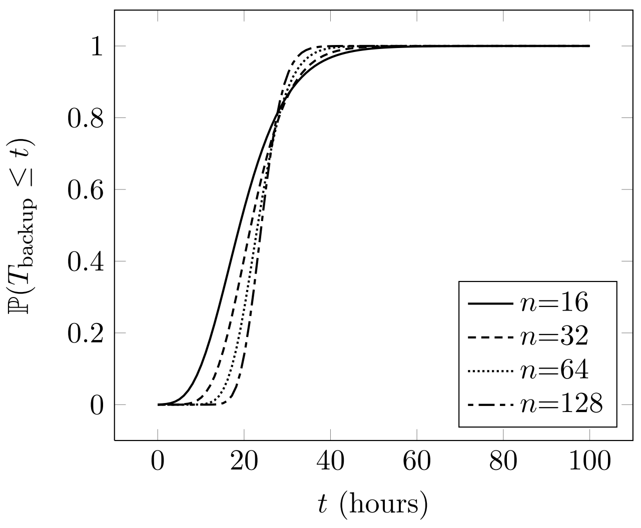


Figure 8: Distribution of the partial-backup duration (time before a proportion 15/16 of the n blocks are stored on peer's machines), to be added to the constant component $\frac{ns}{d_u}$. (Parameters: $\lambda = 1/12$ hours^{-1}, $\mu = 1/12$ hours^{-1}.)

backup times in addition to improving data availability. Remark also that as n increases, the variance of the backup time decreases due to the law of large numbers (the distribution function is steeper).

4.2.3. Restore duration

We consider here a restore request, that consists in gathering a minimum of k blocks among the n peers where a block has been stored. The reasoning is actually very similar to the one followed to obtain (3), since we focus on the time until at least k peers have been seen online. This leads to (still ignoring the transmission duration, that is $\frac{s}{d_u}$ if the storing peers do not have other uplink traffic):

$$\mathbb{P}(T_{\text{restore}} > t) = \sum_{i=n-k+1}^{n} \underbrace{\binom{n}{i} \left(\frac{\lambda}{\lambda+\mu}\right)^{i} \left(\frac{\mu}{\lambda+\mu}\right)^{n-i}}_{\mathbb{P}(i \text{ peers among } n \text{ are offline})} \Phi^{i}_{n-k+1}(t),$$

with $\Phi^i_j(t)$ given in (4).

5. A comparison framework for backup schemes

In this section, we detail the simulation framework we have built in order to assess the performances of our backup scheme, and perform simulations.

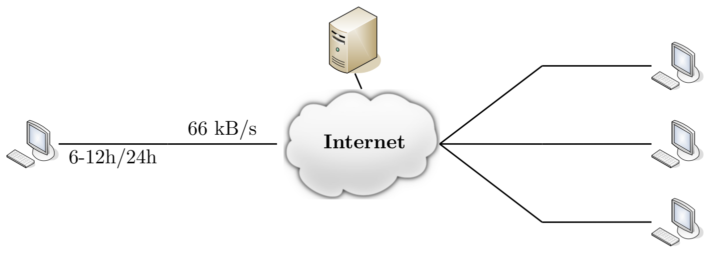


Figure 9: A global picture of the network connecting peers and CDN, as used in [5]. Note that the CDN (Server) has an infinite capacity and 100% availability. However, since the bandwidth used at the CDN is billed, the CDN is not used as a *relay* but as another kind of end-storage (*i.e.,* it does not upload content to other peers but only stores content temporarily until the backing up peers have uploaded content to enough peers.)

As compared to the model introduced in previous section, this allows us to *(i)* extend performance analysis by using additional metrics, that are difficult to capture with a theoretical model. *(ii)* As it well understood that exponential laws are sufficient for system dimensioning, but does not capture fine-grained reality, this simulator takes as an input a probability distribution that has been shown to fit well real availability traces. Finally, it allows to *(iii)* fairly compare our approach to its direct competitors, within the same framework and using the same metrics.

5.1. Competitors

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

5.1.1. P2P *system*

The vast majority of peer-to-peer storage protocols historically presents a purely decentralized system with one-to-one uploads/downloads, without servers [8, 9]. 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.

5.1.2. CDNA *system*

A possible enhancement consists in introducing a CDN to mitigate the low availability of peers [5]. *CDNA* then stands for CDN-assisted. The actual CDN is a central service in the core network, having unbounded capacity. We consider the most representative peer-to-peer variant of the protocol in [5] (*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 schematic view is given in Figure 9. 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 to the peers so that the targeted availability is guaranteed, as stated in (5):

$$\sum_{i=k-s}^{t} \binom{t}{i} \bar{p}^{i} (1-\bar{p})^{t-i} > A_{target}, \tag{5}$$

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

5.1.3. GW&CDN *system*

For completeness, we enhance [5] with our concept of active gateways. This aggregate of two approaches is intended to provide an upper bound on performances, in a realistic deployment scenario. *GW&CDN* then stands for system with gateways and a CDN.

5.2. Parameters and data sets

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

5.2.1. Peer availability

In order to model the up-time of personal computing devices (*i.e.,* peers), we rely on four availability traces, representing four stability categories:

- **Jabber** Provided by the authors of [5], the Jabber trace represents a highly volatile system (instant messaging). The average availability of its devices is 27%, based on the behavior of 10,000 users. By using the FTA toolbox, we extracted the following parameters for modeling this trace, using the Log-Normal distribution: $mean = 1.16$ and $std.dev. = 3.25$ (and $mean = 1.64$ and $std.dev. = 3.54$ for unavailability).

- **Skype** In this trace [25] of about 1,269 peers, the average availability of peers is around 50%, which represents a medium availability when considering peer-to-peer systems (distribution parameters given are in paper [26]).
- **Microsoft** The Microsoft availability dataset [29] contains uptime of around 50,000 desktop PCs, for an average uptime of 80%.
- **DNS** Finally, a highly stable trace is the one constituted by around 60,000 DNS servers [30], for an average availability of 98%.

Characteristics of the last three traces are analyzed in depth in paper [26].

5.2.2. Gateway availability

To model the up-time of residential gateways, we rely on our gateway trace presented in Section 3. We use the parameters extracted though the Failure Trace Archive toolbox to parametrize the best fitting law, which is Log-Normal (with parameters indicated in Section 3). 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: we rely on the gateway trace only for gateway to gateway communication.

5.2.3. 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_{target}, 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_{target}. In the evaluation section, settings are $k = 16$, $A_{target} = 0.7$, and n values are derived from expressions (1) and (5).

5.3. Evaluation

We have implemented an event-based simulator for comparing all approaches in different scenarios. We first present the simulation protocol, before presenting quality metrics for assessing system performances. We then validate this simulator against the theoretical model, before we present actual simulations.

5.3.1. Simulation protocol

We set the simulation with $N = 1,000$ nodes running the same protocol (*P2P*, *CDNA*, *GWA* or *GW&CDN*). Each node alternates between up and down sessions. When a node becomes available, it performs its uncompleted backup operations and restore queries; we then take three random values : $T_{down}, T_{backup}, T_{restore}$. T_{down} follows the user availability law. We consider that each node performs three backups a month while performing only one restore a month. We model this behavior with a Poisson process.

If we have $T_{backup} \leq T_{down}$, a backup is performed. In this case, we randomly choose a file size according to a normal distribution centered on 1GB file with a 128MB standard deviation. Then backup blocks are sent to n among N uniformly chosen other nodes (n corresponding to the number of redundancy blocks). Blocks are transferred at a constant 1.2MB$*$s^{-1} rate.

In the same way, if $T_{restore} \leq T_{down}$ and there are completed backups, a restore query is processed. In this case, we uniformly choose a backup among the completed ones to restore it. Then, the node contacts the backup hosts which in turn send back the data. A restore is completed when the first k blocks are received by the current node.

Finally, a node becomes unavailable after a random delay following the considered unavailability law. In addition, for protocols *GWA* and *GW&CDN*, random delays are chosen for the availability of the gateways: this delay follows a Log-Normal distribution derived from the availability of residential gateways presented in Section 3.

5.3.2. Quality metrics

We evaluate the three systems according to the following four metrics:

5.3.3. 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 or 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 for each system. The time to backup is evaluated as the difference between the time the n^{th} block has been uploaded and the time of the backup request.

5.3.4. Time to offload

A variant measure for *CDNA*, *GWA* and *GW&CDN* 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*-peers and *GWA*-peers respectively.

5.3.5. *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.

5.3.6. *Data buffered*

It describes the size of the buffer that is required on gateways, and is of interest for dimensioning purposes.

5.3.7. *Simulator validation*

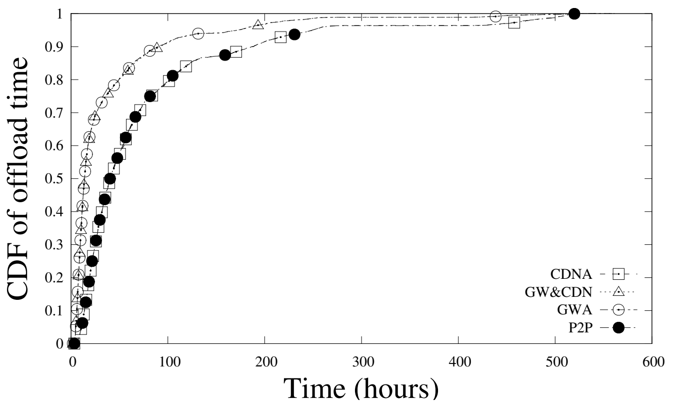


Figure 10: TTB for the four considered backup systems. Host availabilities are modeled using an Exponential distribution, allowing for comparison with the Markov Model results presented in Section 4.

We begin our evaluation by validating the simulator we developed, running it with the same input used for the model presented in Section 4. An exponential distribution for peer availability is used, like it was also used for plotting Figure 7. Results are plotted on Figure 10. The curve to consider on Figure 10 for comparison with the system modeled in Section 4 is the *P2P* curve, as transfer time between a gateway and its peer has been assumed negligible in the model. We observe that the TTB for 90% of the data is around 100 hours on Figure 10, while full completion is around 100 hours on Figure 7; this is a good fit considering the simplifying assumptions used in the model Section.

Additionally, we remark that leveraging the gateways in the simulator (as seen on the *GWA* and *GW&CDN* curves), causes reduced TTB as compared to the *P2P* and *CDNA* systems.

In the rest of this section, we simulate the different scenarios using a Log-Normal distribution, as it has been shown in paper [26] to fit very well existing availability traces, including the Skype, Microsoft and DNS traces.

5.3.8. Simulations for TTB

Figures 11, 12, 13 and 14 present results for TTB for the four availability classes on the left, while time to offload is presented on the right. First learning is that TTB, for *CDNA*, *GWA* and *GW&CDN*, are nearly immediate. This is due to the fact that near-always up hosts are in front of backup requests for storing the blocks, in addition to the fact that more stable systems need less redundancy than volatile ones, which turns into less blocks to upload in order to reach the availability target. On the contrary, backup is significantly longer on the *P2P* system, especially for the two less available traces. This confirms the need to introduce stable third parties for providing efficient backup systems.

The second learning is that when we consider full offload to peers in all systems (Figures on the right), gateway-assisted approaches outperform both *P2P* and *CDNA*[4]. In fact, both *P2P* and *CDNA* suffer from the low upload capacities combined with the transient nodes presence, while in *GWA* and *GW&CDN*, the upload is carried by the gateway thus masking the uploading node unavailability. In the Skype-based system (Figure 13), this translates by a TTB decreasing from days to few hours, as *e.g.,* 600 hours for *CDNA* and only 32 hours for *GWA* (90-th percentile).

5.3.9. Simulations for TTR

Results are consistent and similar for all four availability classes, we then just present results on the Skype trace on Figure 15. *GWA* and *GW&CDN* systems slightly improve upon *P2P* and *CDNA*, due to the fact that a restore

[4]Results are consistent with results in paper [5], where 40 days are necessary to upload 1GB on a 27% mean availability trace (90-th percentile), while we measure 25 days (60 hours) on the 50% Skype trace in our simulations.

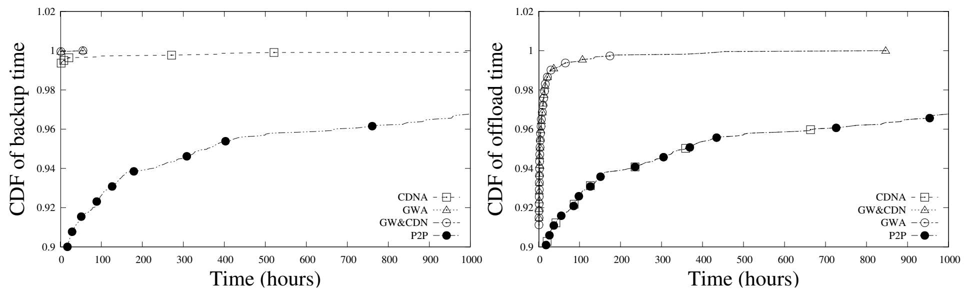


Figure 11: Backup time for the system based on DNS availability; TTB (left) and time to offload on peers (right).

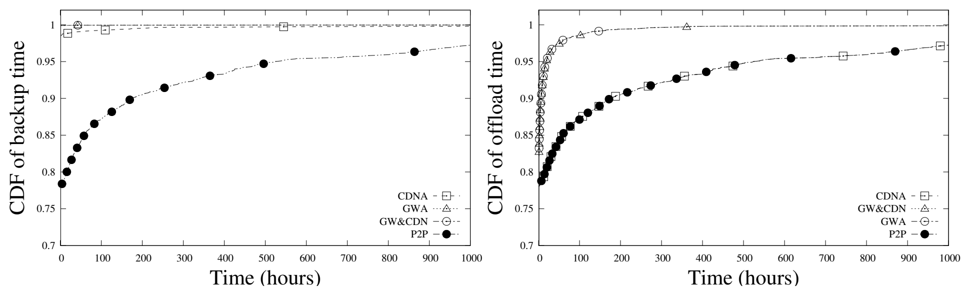


Figure 12: Backup time for the system based on Microsoft availability; TTB (left) and time to offload on peers (right).

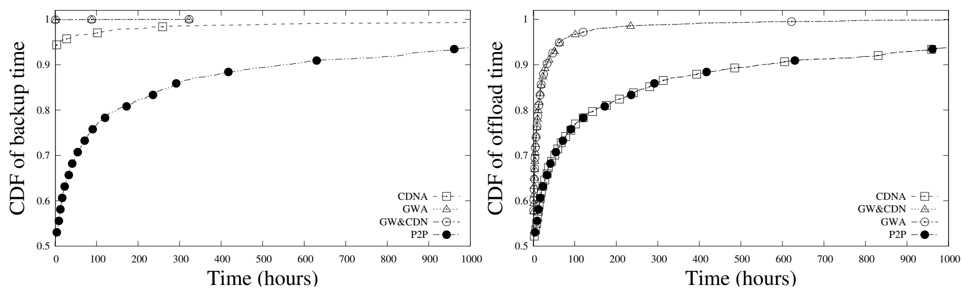
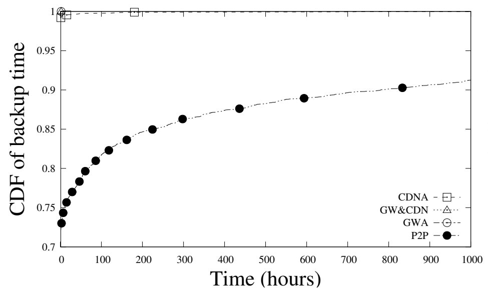


Figure 13: Backup time for the system based on Skype availability; TTB (left) and time to offload on peers (right).

is considered complete when blocks have been gathered on the home gateway of the requesting node, regardless of its online or offline presence.

For TTR, as well for TTB, we remark that there is no substantial improvement to be awaited when implementing the *GW&CDN*, as compared to *GWA*. The gateways are available enough to provide a stable layer between more volatile peers, and no centralized server is mandatory in this



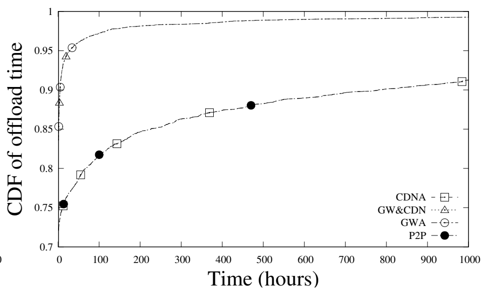


Figure 14: Backup time for the system based on Jabber availability; TTB (left) and time to offload on peers (right).

setting. This is an interesting result for arguing on the relevance of fully decentralized systems for efficient and price-competitive storage solutions.

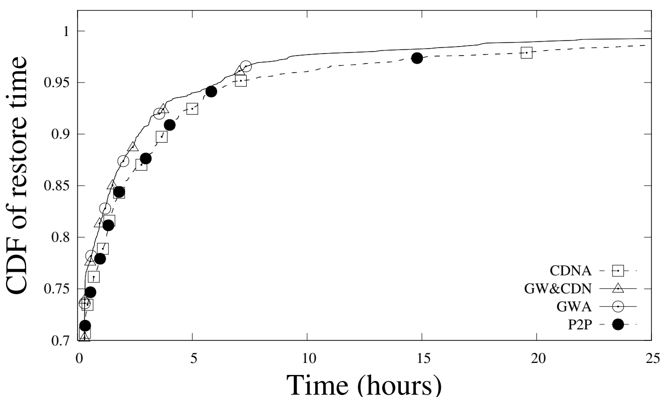


Figure 15: Restore time (TTR) for the system based on Skype availability (similar results for DNS and Microsoft traces).

5.3.10. Simulations for amount of data buffered

It is of interest to study the need for storage space on gateways that is required to implement our proposal. Figure 16 plots the CDF of the maximum amount of data stored at any peer in the *GWA* system, while considering the four availability classes. The more stable the system is, the less storage space has to be provisioned. Indeed, as chosen remote nodes for storage (or requesting nodes) are online most of the time, backup or restore requests do not stack up within the gateway buffer, then reducing the maximum block queue on the disk. The curve presents an experiment with an unbounded disk capacity; in practice a selected node would refuse a

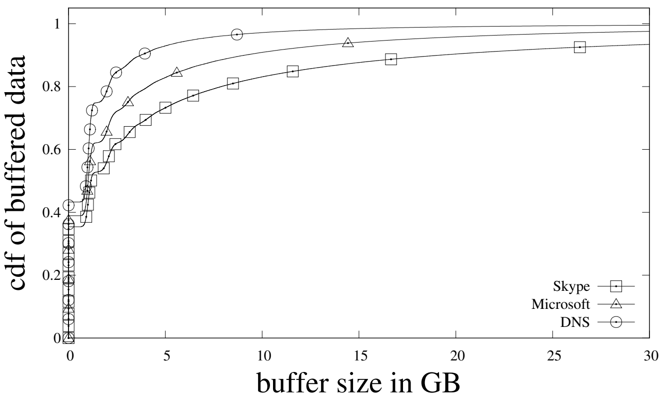


Figure 16: Amount of buffered data at the gateways, for the *GWA*.

storage request for a block if buffer is full, then making it possible to control the actual size of the buffer to be shipped within gateways. In this setting, 10GB disks would suffice to implement most use cases.

Archive size (S)	10 MB	100 MB	1 GB	10 GB
Time to backup	10.46s	1.81mn	32.23h	26.50d

Figure 17: Time to backup in the gateway-assisted system, for 90-th percentile on the Skype trace, for various archive sizes.

We now make archive size vary over one of the less available systems, Skype. Results are reported for the 90-th percentile of completed backups, over a full system simulation run, and are presented on Figure 17. Those results show a linear time increase, as a function of the archive size.

We do not vary the size of the network, N. This is because the time to process a backup or a restore does not depends on N but only on n, the number of nodes where blocks are stored; N does not have in practice a significant impact on the overall system when it is large.

6. Discussion

Our results clearly indicate that our proposal efficiently distributes the centralized buffer scheme of [5], 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 [31]. 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 at night). A similar advantage, appealing for Internet and service providers [32], 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) [22, 23]; 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 [33], 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.

7. Related Work

We compared our approach to the peer-assisted one presented in [5] 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 [17], 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 [34]) 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 uses synthetic traces, 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 [8, 9, 10, 11, 12]. 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 known availability traces. Finally, FS2You [18] 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 [35] or [21]), which avoids the user to let her computer powered on [19]. Another example is the concept of Nano Data Centers [7], 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 [20, 36] and IPTV [37]. Yet, those applications fully run on gateways while, in our approach, the gateway only acts as buffering stage.

8. 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 show that this architectural paradigm shift, significantly improves the user experience of backup systems over previous approaches, while remaining scalable and bear comparison with over-provisioned CDN servers.

References

[1] W. J. Bolosky, J. R. Douceur, D. Ely, , 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, 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, M. Sánchez-Artigas, Availability and Redundancy in Harmony: Measuring Retrieval Times in P2P Storage Systems, in: P2P, 2010.

[4] T. Mager, E. Biersack, P. Michiardi, A measurement study of the wuala on-line storage service, in: P2P, 2012.

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

[6] N. Daswani, H. Garcia-Molina, B. Yang, Open problems in data-sharing peer-to-peer systems, in: ICDT, 2002.

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

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

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

[10] L. P. Cox, C. D. Murray, B. D. Noble, Pastiche: making backup cheap and easy, SIGOPS Oper. Syst. Rev. 36 (2002) 285–298.

[11] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, M. Isard, A cooperative internet backup scheme, in: Usenix ATC, 2003.

[12] M. Landers, H. Zhang, K.-L. Tan, Peerstore: Better performance by relaxing in peer-to-peer backup, in: P2P, 2004.

[13] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, K. Ramchandran, Network coding for distributed storage systems, in: IEEE Transactions on Information Theory, vol. 56, no. 9, 2010.

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

[15] K. Tati, G. M. Voelker, On Object Maintenance in Peer-to-Peer Systems, in: IPTPS, 2006.

[16] P. Maille, L. Toka, Managing a Peer-to-Peer Data Storage System in a Selfish Society, Selected Areas in Communications, IEEE Journal on 26 (7) (2008) 1295 –1301.

[17] A. Gharaibeh, M. Ripeanu, Exploring data reliability tradeoffs in replicated storage systems, in: HPDC, 2009.

[18] F. Liu, Y. Sun, B. Li, B. Li, X. Zhang, FS2You: Peer-Assisted Semipersistent Online Hosting at a Large Scale, IEEE Trans. Parallel Distrib. Syst. 21 (2010) 1442–1457.

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

[20] J. Muñoz Gea, A. Nafaa, J. Malgosa-Sanahuja, T. Rohmer, Design and analysis of a peer-assisted vod provisioning system for managed networks, Multimedia Tools and Applications (2012) 1–36.

[21] I. Kelé andnyi, A. Ludá andnyi, J. Nurminen, Using home routers as proxies for energy-efficient bittorrent downloads to mobile phones, Communications Magazine, IEEE 49 (6) (2011) 142–147.

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

[23] R. Bhagwan, S. Savage, G. Voelker, Understanding Availability, in: IPTPS, 2003.

[24] M. Dischinger, A. Haeberlen, K. P. Gummadi, , S. Saroiu., Characterizing Residential Broadband Networks, in: IMC, 2007.

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

[26] D. Kondo, B. Javadi, A. Iosup, D. Epema, The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems, in: CCGrid, 2010.

[27] OECD broadband statistics, http://www.oecd.org/sti/broadband/.

[28] W. K. Lin, D. M. Chiu, Y. B. Lee, Erasure code replication revisited, in: P2P, 2004.

[29] W. J. Bolosky, J. R. Douceur, D. Ely, M. Theimer, Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs, in: SIGMETRICS, 2000.

[30] J. Pang, J. Hendricks, A. Akella, R. De Prisco, B. Maggs, S. Seshan, Availability, usage, and deployment characteristics of the domain name system, in: IMC, 2004.

[31] T. Leighton, Improving performance on the internet, Commun. ACM 52 (2) (2009) 44–51. doi:10.1145/1461928.1461944.
URL `http://doi.acm.org/10.1145/1461928.1461944`

[32] N. Laoutaris, G. Smaragdakis, P. Rodriguez, R. Sundaram, Delay tolerant bulk data transfers on the internet, in: SIGMETRICS, 2009.

[33] K. Huguenin, E. Le Merrer, N. Le Scouarnec, G. Straub, Hoop: HTTP POST offloading from user devices onto residential gateways, in: ICWS, 2014.

[34] Amazon web services, http://s3.amazonaws.com.

[35] Fon, http://corp.fon.com.

[36] V. Janardhan, H. Schulzrinne, Peer assisted VoD for set-top box based IP network, in: P2P-TV, 2007.

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