Batch utilization

György Barabás & Viktor Trón

1 Introduction

When pushing content to the Swarm network, uploaders are required to attach an attestation of storage rent prepayment to each chunk they post. The latter is essentially a wallet registered on the postage contract seeded with a balance from which storage rent is deduced by the network-wide incentive system. Since this is reminiscent of buying a batch of postage stamps and attaching one to each envelope to be posted, the attestations are called *postage stamps* and the registered wallet a *postage batch.* One can think of batches as collections of *storage slots.* The size of a batch is the number of storage slots and is always specified as a power of 2, with the exponent called *batch depth.* Each slot can hold at most one chunk. Putting a chunk into a slot is like issuing a postage stamp.

In practice, the attached postage stamps are digital signatures which associate the address of a chunk with a *storage slot reference.* This in turn is composed of 1) a reference to the wallet through its *batch ID,* and 2) a *within-batch index.* The fact that each slot can hold at most one chunk ensures that batches cannot issue more stamps than the volume registered with them. However, for an *overissuance* incident to be detectable locally by storer nodes, the within-batch indices are arranged such that the highest ν bits match the prefix of the chunk they are assigned to. These ν bits define $n = 2^\nu$ *buckets*, the other half of the index is essentially a counter within the buckets, which is sequentially assigned to chunks. If the batch depth is d and there are $n = 2^\nu$ buckets, then each bucket will hold a maximum of $2^{d-\nu}$ chunks. $\kappa = d - \nu$, the log size of a bucket is called *bucket depth.* The bucket size $k = 2^\kappa$ provides an exclusive upper bound to within-bucket indices. This enforces a uniformity of stamp issuance across the 2^ν buckets, therefore ν is called *uniformity depth.*

Overusing a batch is now easily detected by any storer node as long as their storage depth is shallower than the batch's uniformity depth. In that case, each bucket of a batch is entirely within the node's reserve. Overissuance is therefore immediately caught, since the multiple chunks assigned to the same slot index are seen by any node in that neighbourhood.

When all slots are filled, we say the batch is *fully utilized.* Although the *a priori* distribution of chunks is uniform (and therefore the expected number of chunks falling into each bucket is the same), their stochastic assignment means that there is necessarily some variance. It is practically impossible to fully fill all buckets before eventually attempting to add to one that is already full. We assume that the uploader is unable to affect the address of the chunks (unencrypted fixed content) so in this scenario they are unable to continue uploading. Because of this, one may legitimately consider the batch to be no longer usable. The number of stamps hitherto issued by the batch is called its *effective batch size,* and its ratio to the batch size its *batch utilization rate.*

Below we explore the effect of batch parameters on their utilization rate. With an insight into utilization rates as a function of the number of buckets $n = 2^\nu$ and the size of buckets $k = 2^\kappa$, we will have a way to calibrate the expected effective batch size to be presented to users in the context of a batch purchase user experience.

2 The effective utilization rate

The problem of assigning chunks to storage slots is analogous with the process of throwing marbles, one after the other, in boxes which are initially empty. Each throw may end up in any of the boxes with equal probability, and thus the marbles get distributed across the boxes more or less evenly through time. However, the time will come when the boxes start filling up. At that point, a marble may by chance end up being thrown into a box that is already full, and thus get rejected. Substituting chunks for marbles, buckets for boxes, and the act of signing a stamp for throwing a marble, we recover our original scenario. Marbles ending up in a box with equal probability corresponds to the fact that a random chunk has equal chance of being assigned to a bucket since the hash function has uniform distribution. Repeated rounds of marble throwing correspond to consecutive stamping of multiple chunks of the uploaded content; rounds constitute repeated independent trials.

Taking a particular bucket, each round of stamping is a "success" if the stamp falls into the bucket, and a "failure" otherwise. Due to the fact that a marble may end up in each box with equal chance, the probability of success is $1/n$ and the probability of failure is $1 - 1/n$. The number of stamps issued to the bucket after a given number of rounds is described by the negative binomial distribution $\mathcal{B}(k, 1/n)$, where the first parameter k is the number of failed rounds before we stop counting, and the second parameter is the probability of success.

The negative binomial distribution $\mathcal{B}(k, 1/n)$ is equivalent to the sum of k independent geometrically distributed variables with parameter $1/n$. If k is sufficiently large, the central limit theorem ensures that this sum converges to a normal distribution. Formally, if $\mathcal{Y}_i(1/n)$ are independent geometrically distributed variables with parameter $1/n$, and $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2, then for large k we have

$$\mathcal{B}(k, 1/n) = \sum_{i=1}^{k} \mathcal{Y}_i(1/n) \approx \mathcal{N}(kn, kn(n-1)) \tag{1}$$

The reason for the above form of the mean and variance in the normal distribution is as follows: given a single geometrically distributed variable with parameter $1/n$, its mean is n and its variance is $n(n-1)$. When adding these up over k independent variables, we arrive at $\mu = kn$ and $\sigma^2 = kn(n-1)$ in the limiting normal distribution. Figure 1 shows how this normal approximation performs, for various values of k. The fit is generally good, except when k and the number of stamps are simultaneously low.

The negative binomial distribution estimates the number of rounds a particular bucket fills up, given its size k and the number of buckets n. The rounds of stamping can be conceived of as parallel independent attempts at filling up buckets. Stamps issued for failed rounds still end up in one of the other $n-1$ buckets, and therefore the same probability variable counts the total stamps issued by the batch in all buckets at the time the first one fills up. Thus, we want to know how the *minimum* of n independent negative binomial variates, $X = \min_{i \in \{1 \ldots n\}} \mathcal{B}(k, 1/n)$, is distributed.

This so-called *extreme value distribution* would give us the distribution of the absolute number of total rounds needed until the first event when one of the buckets gets full. Since this is when the batch is considered effectively used up, this distribution can be used to calculate the effective utilization rate of the batch.

The extreme value distribution corresponding to the minimum of n independent negative binomial variates $\mathcal{B}(k, 1/n)$ can be obtained via two methods: either by numerical simulation, or by using the normal approximation of Eq. 1 to get the minimum of n normal variates $X = \min_{i \in \{1 \ldots n\}} \mathcal{N}(kn, kn(n-1))$ instead. In deriving the latter, we rely on the fact that the extreme value distribution for the minimum of n independent variates drawn from the standard

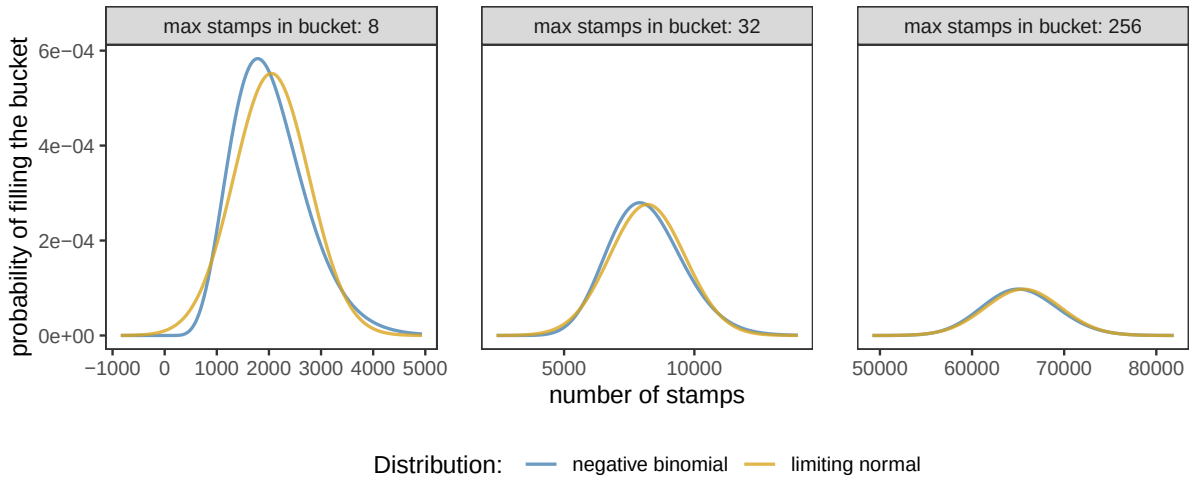

Figure 1: Approximating negative binomial distributions with limiting normal ones, for different values of the maximum number of stamps k that fit in a bucket (panels). The number of buckets n is fixed at $2^8 = 256$. Overall, this approximation is robust even for low values of k. However, the quality of the fit is poor when k and the number of stamps are simultaneously low. In that case, the limiting normal distribution can even reach out into the meaningless negative region. In those cases, the normal approximation should be avoided. By contrast, for larger values of k the normal approximation becomes excellent everywhere.

normal distribution (with zero mean and unit variance) is known. It is the Gumbel distribution, which reads

$$\mathcal{G}(x;\alpha,\beta) = \frac{1}{\beta}\exp\left[\frac{x+\alpha}{\beta} - \exp\left(\frac{x+\alpha}{\beta}\right)\right]. \tag{2}$$

Here x is the independent variable (in our case, the number of rounds of stamping), and α and β are the location and scale parameters, respectively.[1] They are in turn given by

$$\begin{aligned} \alpha &= \Phi^{-1}\left(1 - \frac{1}{n}\right), \\ \beta &= \Phi^{-1}\left(1 - \frac{\mathrm{e}^{-1}}{n}\right) - \alpha, \end{aligned} \tag{3}$$

where $\mathrm{e}^{-1} = \exp(-1) \approx 0.368$, n is the number of normal variates whose minimum we are looking for, and $\Phi^{-1}(\cdot)$ is the the inverse of the error function. The mean and standard deviation of the Gumbel distribution are, respectively, $\alpha + \gamma\beta$ and $\beta\pi/\sqrt{6}$, where $\gamma \approx 0.5772$ is the Euler–Mascheroni constant. Its quantile function (which we will make use of below) is analytically expressible, and reads

$$Q(p;\mathcal{G}) = \beta\,\log(-\log(1-p)) - \alpha, \tag{4}$$

where $0 < p < 1$ is a probability quantile.

In our case, the normal distributions whose minimum we are looking for is not standard, but has mean $\mu = kn$ instead of zero, and variance $\sigma^2 = kn(n-1)$ instead of one. Therefore, the Gumbel distribution needs to be appropriately rescaled. Denoting this scaled probability

[1]The Gumbel distribution is often given using the convention that one is looking for the maximum of n normal variates, instead of their minimum. One can change between the two by simply flipping the sign of x.

distribution by $\mathcal{X}$, we have

$$\mathcal{X}(x;\mu,\sigma,\alpha,\beta) = \frac{1}{\sigma}\mathcal{G}\left(\frac{x-\mu}{\sigma};\alpha,\beta\right) \tag{5}$$

(the overall factor of $1/\sigma$ restores the proper normalization of the scaled function). Consequently, the mean also gets rescaled to $\mu - \sigma(\alpha + \gamma\beta)$, the standard deviation to $\sigma\beta\pi/\sqrt{6}$, and the quantile function to

$$\begin{aligned} Q(p;\mathcal{X}) &= \mu - \sigma[\alpha - \beta\,\log(-\log(1-p))] \\ &= kn - \sqrt{kn(n-1)}\left[\Phi^{-1}\left(1-\frac{1}{n}\right) - \Phi^{-1}\left(1-\frac{\mathrm{e}^{-1}}{n}\right)\log(-\log(1-p))\right], \end{aligned} \tag{6}$$

where we used $\mu = kn$, $\sigma^2 = kn(n-1)$, and Eq. 3 to express α and β.

As a final step, we normalize this function by the product kn. Doing so gives the quantile of stamps per bucket per chunk (since every bucket can hold k chunks), and is precisely the effective utilization rate we set out to obtain (or, rather, its pth quantile). Denoting this by U_{eff}, we have

$$U_{\text{eff}} = \frac{Q(p;\mathcal{X})}{kn} = 1 - \sqrt{\frac{2^\nu - 1}{2^{\kappa+\nu}}}\left[\Phi^{-1}\left(1-\frac{1}{2^\nu}\right) - \Phi^{-1}\left(1-\frac{\mathrm{e}^{-1}}{2^\nu}\right)\log(-\log(1-p))\right], \tag{7}$$

where $\kappa = \log_2(k)$ and $\nu = \log_2(n)$, as before.

The meaning of the quantile function is that only a fraction p of variates are smaller than or equal to $Q(p;\mathcal{X})$. For example, $Q(0.001;\mathcal{X})$ is the number of stamps such that only one in a thousand (0.1%) of users will get their first bucket filled with fewer stamps. This can be used to protect Swarm users from being surprised at the effective utilization of their postage batches. We therefore calibrate batch volumes using the above $p = 0.001$ as a reasonable worst-case scenario. That is, given n and k, we do not say that the user now has 2^{n+k} chunks of data at their disposal. Instead, we multiply this theoretical maximum volume by U_{eff} after setting $p = 0.001$, and report that value as the amount of available storage space.

As mentioned before, the approximation of Eq. 7 breaks down when k and p are simultaneously small. Since we decided to fix $p = 0.001 \ll 1$, we must turn to numerical simulations for estimating the normalized quantile function for smaller values of k. The simulation procedure is simple: we generate n negative binomially distributed random variates $\mathcal{B}(k, 1/n)$ and record their minimum, then repeat this 100 000 times. Using these 100 000 minima, we sort them in increasing order and read off the value of the 1000th result to obtain the 0.1% quantile. Finally, we normalize the outcome by kn. Figure 2 shows both the simulated and analytically approximated results. As a rule of thumb, Eq. 7 yields a good approximation for $\kappa \geq 8$.

We use simulated results for $0 \leq \kappa \leq 10$ and Eq. 7 for $\kappa \geq 11$ to generate effective utilization rates (Table 1). The table provides these values for uniformity depths $\nu = 12$ and $\nu = 16$, and log bucket sizes κ ranging from 0 to 25. Above this bucket size, the effective utilization rate is already above 99.9%, which we consider large enough to justify no subsequent calibrations on the theoretical volumes.

3 Packed address chunks and erasure coding

In principle, we now have all the tools to compute how much space a user can be expected to utilize: we take ν and κ, look up their combination in Table 1, and multiply the corresponding volume by U_{eff} in the same row. For example, given a uniformity depth of $\nu = 16$ (assumed to

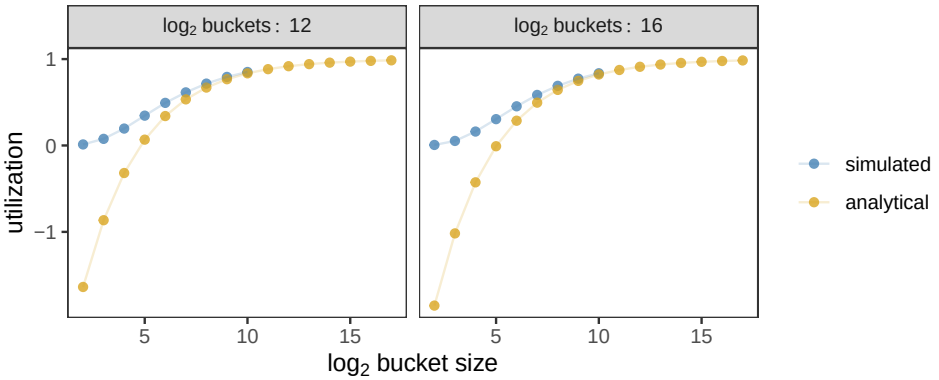

Figure 2: Effective utilization rates (ordinate) against log bucket size κ (abscissa) for different values of uniformity depth ν (12 in the left and 16 in the right panel). Blue points, estimated up to $\kappa = 10$, are simulated effective utilization rates (0.1% quantiles, normalized by $2^{\nu+\kappa}$). Yellow points are calculated from the analytical approximation of Eq. 7. The points are connected by lines for visual clarity. For small κ, the analytical approximation breaks down, so one must use the simulated results. The match between the approximation and the simulations is already good for $\kappa = 8$, and they are practically indistinguishable for $\kappa = 10$.

be a system-wide setting) and a user who bought postage stamps at a batch depth of $\kappa = 11$, the storage space guaranteed would not be the theoretical maximum volume of 549.76 GB, but $549.76\,\text{GB} \cdot 87.39\% = 480.43\,\text{GB}$ instead. However, this is still an overestimate of the amount of data the user will be able to upload due to two factors: packed address chunks (PACs) and erasure coding.

Swarm represents files via the Swarm hash tree—a Merkle tree with a branching factor of either 128 (unencrypted files) or 64 (encrypted files). Only the leaves of the tree store the original data; the nodes contain chunk references (plus an encryption key for encrypted files). Those nodes are the PACs. It is easy to calculate what fraction of a Swarm file is made up by them. Let the branching factor be b (either 64 or 128, depending on the presence or absence of encryption), the number of PACs be P, and the number of chunks in the file (without PACs) be N. The tree structure means that, at every level, another factor of $1/b$ multiplies the fraction of chunks that are PACs. Mathematically,

$$P = N\left(\frac{1}{b} + \frac{1}{b^2} + \frac{1}{b^3}\ldots\right), \tag{8}$$

with the sum having as many terms as the number of levels in the tree. In practice, we can give an upper bound to the fraction of PACs by assuming that the number of levels is formally infinite. This turns the above sum into an infinite geometric series, which converges very rapidly due to b being large. Using the summation formula for an infinite geometric series, we obtain

$$P = N\sum_{k=1}^{\infty}\frac{1}{b^k} = \frac{N}{b-1}. \tag{9}$$

The total number of chunks is therefore $N + P = N + N/(b-1) = Nb/(b-1)$. That is, one has an overhead of $b/(b-1)$ for each file, on top of its original size. This overhead is 128/127 (unencrypted case) or 64/63 (encrypted case).

Continuing with our previous numerical example of $\nu = 16$ and $\kappa = 11$: assuming that the files are unencrypted, the effectively usable volume is not simply $549.76\,\text{GB} \cdot 87.39\% = 480.43\,\text{GB}$,

ν	κ	volume	U_{eff}
12	0	16.78 MB	0%
12	1	33.55 MB	0.02%
12	2	67.11 MB	1.2%
12	3	134.22 MB	7.67%
12	4	268.44 MB	19.58%
12	5	536.87 MB	34.53%
12	6	1.07 GB	49.31%
12	7	2.15 GB	61.48%
12	8	4.29 GB	71.61%
12	9	8.59 GB	79.35%
12	10	17.18 GB	85.06%
12	11	34.36 GB	88.34%
12	12	68.72 GB	91.76%
12	13	137.44 GB	94.17%
12	14	274.88 GB	95.88%
12	15	549.76 GB	97.09%
12	16	1.10 TB	97.94%
12	17	2.20 TB	98.54%
12	18	4.40 TB	98.97%
12	19	8.80 TB	99.27%
12	20	17.59 TB	99.48%
12	21	35.18 TB	99.64%
12	22	70.37 TB	99.74%
12	23	140.74 TB	99.82%
12	24	281.47 TB	99.87%
12	25	562.95 TB	99.91%

ν	κ	volume	U_{eff}
16	0	268.44 MB	0%
16	1	536.87 MB	0.01%
16	2	1.07 GB	0.63%
16	3	2.15 GB	5.26%
16	4	4.29 GB	16.14%
16	5	8.59 GB	30.48%
16	6	17.18 GB	45.33%
16	7	34.36 GB	58.49%
16	8	68.72 GB	69.02%
16	9	137.44 GB	77.37%
16	10	274.88 GB	83.59%
16	11	549.76 GB	87.39%
16	12	1.10 TB	91.08%
16	13	2.20 TB	93.7%
16	14	4.40 TB	95.54%
16	15	8.80 TB	96.85%
16	16	17.59 TB	97.77%
16	17	35.18 TB	98.42%
16	18	70.37 TB	98.89%
16	19	140.74 TB	99.21%
16	20	281.47 TB	99.44%
16	21	562.95 TB	99.61%
16	22	1.13 PB	99.72%
16	23	2.25 PB	99.8%
16	24	4.50 PB	99.86%
16	25	9.01 PB	99.9%

Table 1: Effective utilization rates (U_{eff}) of volumes of postage batches. Here $\nu = \log_2(n)$ is the log number of buckets (1st column of tables), and $\kappa = \log_2(k)$ is the log batch depth (2nd column). The left table is for $\nu = 12$; the right one is for $\nu = 16$. The theoretical maximum number of chunks that can be stored is $2^{\nu+\kappa} = kn$, and since one chunk is 4 KB in size, the maximum volume is $2^{\nu+\kappa} \cdot 4$ KB (3rd column). The calculated utilization rates (4th column) were obtained via numerical simulations for $\kappa \leq 10$, and by using the analytical approximation of Eq. 7 for $\kappa > 10$.

as calculated earlier. Instead, this must additionally be divided by the overhead of 128/127, resulting in 476.68 GB of available space. If the files are encrypted, then the overhead to divide by is 64/63, giving 472.93 GB.

Apart from PACs (which are always used), there is optional erasure coding that users can choose to bolster the retrievability of their files. As elaborated elsewhere,[2] this works by partitioning each 128-chunk sequence (or 64-chunk sequence, for encrypted files) into m "original" data chunks and p parity chunk, such that losing any p of the $m + p$ chunks would still allow one to perfectly reconstruct the original file. Table 2 summarizes m and p for each of the five available erasure levels in Swarm.

[2]Trón et al. (2024) Non-local redundancy: Erasure coding and dispersed replicas for robust retrieval in the Swarm peer-to-peer network. arXiv, 2409.01259, doi: 10.48550/arXiv.2409.01259

security		unencrypted		encrypted	
level	name	chunks	parities	chunks	parities
0	NONE	128	0	64	0
1	MEDIUM	119	9	59	9
2	STRONG	107	21	53	21
3	INSANE	97	31	48	31
4	PARANOID	38	90	19	90

Table 2: Security levels for the five possible levels of erasure coding in Swarm (columns 1-2), with the corresponding compositions of every 128-chunk sequence for unencrypted (columns 3-4) and encrypted (columns 5-6) content. Columns 3 and 4 always sum to 128, as does twice column 5 plus column 6 (with potential rounding error to 127 due to the fact that encrypted chunks take up twice as much space as unencrypted ones).

For the same example as above: assuming that a user uploads unencrypted files at the STRONG erasure level, we see from Table 2 that 21 parity chunks are added to every 107 original data chunks, per each 128-chunk data segment. This means an additional overhead of 128/107 on each file. Thus, the usable effective volume is not $549.76\,\text{GB} \cdot 87.39\% \cdot (127/128) = 476.68\,\text{GB}$, but $549.76\,\text{GB} \cdot 87.39\% \cdot (127/128) \cdot (107/128) = 398.47\,\text{GB}$. This is the final, effectively usable space that can be guaranteed to the user, given $\nu = 16$ and $\kappa = 11$, no encryption, and STRONG erasure levels.

Denoting the PAC overhead by H_p and the erasure overhead by H_e, the formula for the final effective volume V_eff is as follows:

$$V_\text{eff} = 2^{\nu+\kappa} \frac{U_\text{eff}}{H_\text{p} H_\text{e}} \cdot 4\,\text{KB}. \tag{10}$$

Here $kn = 2^{\nu+\kappa}$ is the theoretical maximum volume (in numbers of chunks), U_eff discounts this with the effective utilization rate as calculated in Section 2, and $H_\text{p}H_\text{e}$ further discounts it by taking packed address chunks and erasure coding into account. We multiply by 4 KB in the end, because that is the size of each chunk. Tables 3-6 summarize all effective volumes for $\nu = 12$ (Tables 3 and 5) and $\nu = 16$ (Tables 4 and 6), for unencrypted (Tables 3 and 4) and encrypted (Tables 5 and 6) content, and for batch depths κ ranging from 0 to 25.

κ	volume	NONE	MEDIUM	STRONG	INSANE	PARANOID
0	16.78 MB	0.00 B	0.00 B	0.00 B	0.00 B	0.00 B
1	33.55 MB	8.13 kB	7.56 kB	6.79 kB	6.16 kB	2.41 kB
2	67.11 MB	800.60 kB	744.31 kB	669.25 kB	606.71 kB	237.68 kB
3	134.22 MB	10.21 MB	9.49 MB	8.54 MB	7.74 MB	3.03 MB
4	268.44 MB	52.16 MB	48.49 MB	43.60 MB	39.53 MB	15.48 MB
5	536.87 MB	183.92 MB	170.99 MB	153.75 MB	139.38 MB	54.60 MB
6	1.07 GB	525.29 MB	488.36 MB	439.11 MB	398.07 MB	155.95 MB
7	2.15 GB	1.31 GB	1.22 GB	1.10 GB	992.70 MB	388.89 MB
8	4.29 GB	3.05 GB	2.84 GB	2.55 GB	2.31 GB	906.00 MB
9	8.59 GB	6.76 GB	6.29 GB	5.65 GB	5.13 GB	2.01 GB
10	17.18 GB	14.50 GB	13.48 GB	12.12 GB	10.99 GB	4.30 GB
11	34.36 GB	30.12 GB	28.00 GB	25.18 GB	22.82 GB	8.94 GB
12	68.72 GB	62.56 GB	58.16 GB	52.30 GB	47.41 GB	18.57 GB
13	137.44 GB	128.42 GB	119.39 GB	107.35 GB	97.32 GB	38.12 GB
14	274.88 GB	261.49 GB	243.10 GB	218.59 GB	198.16 GB	77.63 GB
15	549.76 GB	529.57 GB	492.33 GB	442.68 GB	401.31 GB	157.21 GB
16	1.10 TB	1.07 TB	993.32 GB	893.15 GB	809.68 GB	317.19 GB
17	2.20 TB	2.15 TB	2.00 TB	1.80 TB	1.63 TB	638.30 GB
18	4.40 TB	4.32 TB	4.02 TB	3.61 TB	3.27 TB	1.28 TB
19	8.80 TB	8.66 TB	8.05 TB	7.24 TB	6.57 TB	2.57 TB
20	17.59 TB	17.36 TB	16.14 TB	14.52 TB	13.16 TB	5.16 TB
21	35.18 TB	34.78 TB	32.34 TB	29.08 TB	26.36 TB	10.33 TB
22	70.37 TB	69.64 TB	64.74 TB	58.21 TB	52.77 TB	20.67 TB
23	140.74 TB	139.38 TB	129.58 TB	116.52 TB	105.63 TB	41.38 TB
24	281.47 TB	278.92 TB	259.30 TB	233.16 TB	211.37 TB	82.80 TB
25	562.95 TB	558.04 TB	518.81 TB	466.49 TB	422.89 TB	165.67 TB

Table 3: Assuming $\nu = 12$ and unencrypted files, this table lists, for various values of the batch depth κ (1st column), the theoretical maximum volume $2^{\nu+\kappa} \cdot 4$ KB (2nd column) and the effective usable volumes for the five different erasure levels available in Swarm (last five columns). The values in these last five columns were calculated using Eq. 10.

κ	volume	NONE	MEDIUM	STRONG	INSANE	PARANOID
0	268.44 MB	0.00 B	0.00 B	0.00 B	0.00 B	0.00 B
1	536.87 MB	44.70 kB	41.56 kB	37.37 kB	33.88 kB	13.27 kB
2	1.07 GB	6.66 MB	6.19 MB	5.57 MB	5.05 MB	1.98 MB
3	2.15 GB	112.06 MB	104.18 MB	93.68 MB	84.92 MB	33.27 MB
4	4.29 GB	687.62 MB	639.27 MB	574.81 MB	521.09 MB	204.14 MB
5	8.59 GB	2.60 GB	2.41 GB	2.17 GB	1.97 GB	771.13 MB
6	17.18 GB	7.73 GB	7.18 GB	6.46 GB	5.86 GB	2.29 GB
7	34.36 GB	19.94 GB	18.54 GB	16.67 GB	15.11 GB	5.92 GB
8	68.72 GB	47.06 GB	43.75 GB	39.34 GB	35.66 GB	13.97 GB
9	137.44 GB	105.51 GB	98.09 GB	88.20 GB	79.96 GB	31.32 GB
10	274.88 GB	227.98 GB	211.95 GB	190.58 GB	172.77 GB	67.68 GB
11	549.76 GB	476.68 GB	443.16 GB	398.47 GB	361.23 GB	141.51 GB
12	1.10 TB	993.65 GB	923.78 GB	830.63 GB	753.00 GB	294.99 GB
13	2.20 TB	2.04 TB	1.90 TB	1.71 TB	1.55 TB	606.90 GB
14	4.40 TB	4.17 TB	3.88 TB	3.49 TB	3.16 TB	1.24 TB
15	8.80 TB	8.45 TB	7.86 TB	7.07 TB	6.41 TB	2.51 TB
16	17.59 TB	17.07 TB	15.87 TB	14.27 TB	12.93 TB	5.07 TB
17	35.18 TB	34.36 TB	31.94 TB	28.72 TB	26.04 TB	10.20 TB
18	70.37 TB	69.04 TB	64.19 TB	57.71 TB	52.32 TB	20.50 TB
19	140.74 TB	138.54 TB	128.80 TB	115.81 TB	104.99 TB	41.13 TB
20	281.47 TB	277.72 TB	258.19 TB	232.16 TB	210.46 TB	82.45 TB
21	562.95 TB	556.35 TB	517.23 TB	465.07 TB	421.61 TB	165.17 TB
22	1.13 PB	1.11 PB	1.04 PB	931.23 TB	844.20 TB	330.72 TB
23	2.25 PB	2.23 PB	2.07 PB	1.86 PB	1.69 PB	661.97 TB
24	4.50 PB	4.46 PB	4.15 PB	3.73 PB	3.38 PB	1.32 PB
25	9.01 PB	8.93 PB	8.30 PB	7.46 PB	6.77 PB	2.65 PB

Table 4: As Table 3, but with $\nu = 16$.

κ	volume	NONE	MEDIUM	STRONG	INSANE	PARANOID
0	16.78 MB	0.00 B	0.00 B	0.00 B	0.00 B	0.00 B
1	33.55 MB	8.06 kB	7.43 kB	6.68 kB	6.05 kB	2.39 kB
2	67.11 MB	794.30 kB	732.25 kB	657.78 kB	595.72 kB	235.81 kB
3	134.22 MB	10.13 MB	9.34 MB	8.39 MB	7.60 MB	3.01 MB
4	268.44 MB	51.75 MB	47.70 MB	42.85 MB	38.81 MB	15.36 MB
5	536.87 MB	182.48 MB	168.22 MB	151.11 MB	136.86 MB	54.17 MB
6	1.07 GB	521.16 MB	480.44 MB	431.58 MB	390.87 MB	154.72 MB
7	2.15 GB	1.30 GB	1.20 GB	1.08 GB	974.73 MB	385.83 MB
8	4.29 GB	3.03 GB	2.79 GB	2.51 GB	2.27 GB	898.87 MB
9	8.59 GB	6.71 GB	6.19 GB	5.56 GB	5.03 GB	1.99 GB
10	17.18 GB	14.38 GB	13.26 GB	11.91 GB	10.79 GB	4.27 GB
11	34.36 GB	29.88 GB	27.55 GB	24.74 GB	22.41 GB	8.87 GB
12	68.72 GB	62.07 GB	57.22 GB	51.40 GB	46.55 GB	18.43 GB
13	137.44 GB	127.41 GB	117.45 GB	105.51 GB	95.55 GB	37.82 GB
14	274.88 GB	259.43 GB	239.16 GB	214.84 GB	194.57 GB	77.02 GB
15	549.76 GB	525.40 GB	484.35 GB	435.09 GB	394.05 GB	155.98 GB
16	1.10 TB	1.06 TB	977.21 GB	877.84 GB	795.02 GB	314.70 GB
17	2.20 TB	2.13 TB	1.97 TB	1.77 TB	1.60 TB	633.27 GB
18	4.40 TB	4.28 TB	3.95 TB	3.55 TB	3.21 TB	1.27 TB
19	8.80 TB	8.60 TB	7.92 TB	7.12 TB	6.45 TB	2.55 TB
20	17.59 TB	17.23 TB	15.88 TB	14.27 TB	12.92 TB	5.11 TB
21	35.18 TB	34.51 TB	31.81 TB	28.58 TB	25.88 TB	10.24 TB
22	70.37 TB	69.09 TB	63.69 TB	57.22 TB	51.82 TB	20.51 TB
23	140.74 TB	138.29 TB	127.48 TB	114.52 TB	103.71 TB	41.05 TB
24	281.47 TB	276.72 TB	255.10 TB	229.16 TB	207.54 TB	82.15 TB
25	562.95 TB	553.65 TB	510.40 TB	458.49 TB	415.24 TB	164.36 TB

Table 5: As Table 3, but with encrypted content.

κ	volume	NONE	MEDIUM	STRONG	INSANE	PARANOID
0	268.44 MB	0.00 B	0.00 B	0.00 B	0.00 B	0.00 B
1	536.87 MB	44.35 kB	40.89 kB	36.73 kB	33.26 kB	13.17 kB
2	1.07 GB	6.61 MB	6.09 MB	5.47 MB	4.96 MB	1.96 MB
3	2.15 GB	111.18 MB	102.49 MB	92.07 MB	83.38 MB	33.01 MB
4	4.29 GB	682.21 MB	628.91 MB	564.95 MB	511.65 MB	202.53 MB
5	8.59 GB	2.58 GB	2.38 GB	2.13 GB	1.93 GB	765.05 MB
6	17.18 GB	7.67 GB	7.07 GB	6.35 GB	5.75 GB	2.28 GB
7	34.36 GB	19.78 GB	18.24 GB	16.38 GB	14.84 GB	5.87 GB
8	68.72 GB	46.69 GB	43.04 GB	38.66 GB	35.02 GB	13.86 GB
9	137.44 GB	104.68 GB	96.50 GB	86.69 GB	78.51 GB	31.08 GB
10	274.88 GB	226.19 GB	208.52 GB	187.31 GB	169.64 GB	67.15 GB
11	549.76 GB	472.93 GB	435.98 GB	391.64 GB	354.69 GB	140.40 GB
12	1.10 TB	985.83 GB	908.81 GB	816.39 GB	739.37 GB	292.67 GB
13	2.20 TB	2.03 TB	1.87 TB	1.68 TB	1.52 TB	602.12 GB
14	4.40 TB	4.14 TB	3.81 TB	3.43 TB	3.10 TB	1.23 TB
15	8.80 TB	8.39 TB	7.73 TB	6.94 TB	6.29 TB	2.49 TB
16	17.59 TB	16.93 TB	15.61 TB	14.02 TB	12.70 TB	5.03 TB
17	35.18 TB	34.09 TB	31.43 TB	28.23 TB	25.57 TB	10.12 TB
18	70.37 TB	68.50 TB	63.15 TB	56.72 TB	51.37 TB	20.34 TB
19	140.74 TB	137.45 TB	126.71 TB	113.82 TB	103.08 TB	40.80 TB
20	281.47 TB	275.53 TB	254.01 TB	228.18 TB	206.65 TB	81.80 TB
21	562.95 TB	551.97 TB	508.85 TB	457.10 TB	413.98 TB	163.87 TB
22	1.13 PB	1.11 PB	1.02 PB	915.26 TB	828.91 TB	328.11 TB
23	2.25 PB	2.21 PB	2.04 PB	1.83 PB	1.66 PB	656.76 TB
24	4.50 PB	4.43 PB	4.08 PB	3.67 PB	3.32 PB	1.31 PB
25	9.01 PB	8.86 PB	8.17 PB	7.34 PB	6.64 PB	2.63 PB

Table 6: As Table 3, but with $\nu = 16$ and with encrypted content.