Extrapolating network totals from hidden-service statistics

George Kadianakis and Karsten Loesing
asn@torproject.org,karsten@torproject.org

Tor Tech Report 2015-01-001[*]
January 31, 2015

Abstract

Starting on December 19, 2014, we added two new statistics to the Tor software that shall give us some first insights into hidden-service usage. The first statistic is the number of cells on rendezvous circuits observed by a rendezvous point, and the second is the number of unique .onion addresses observed by a hidden-service directory. Each relay that opts in to reporting these statistics publishes these two numbers for 24-hour intervals of operation. In the following, we describe an approach for extrapolating network totals from these statistics. The goal is to learn what amount of traffic can be attributed to hidden-service usage and how many unique .onion addresses exist in the network. We show that we can extrapolate network totals with reasonable accuracy as long as at least 1% of relays report these statistics.

Introduction

As of December 19, 2014, a small number of relays has started reporting statistics on hidden-service usage. Similar to other statistics, these statistics are based solely on what the reporting relay observes, without exchanging observations with other relays. In this report we describe a method for extrapolating these statistics to network totals.

Figure 1 gives an overview of the extrapolation method where each step corresponds to a section in this report. In step 1 we parse the statistics that relays report in their extra-info descriptors. These statistics contain noise that was added by relays to obfuscate original observations, which we attempt to remove in step 2. In step 3 we process consensuses to derive network fractions of reporting relays, that is, what fraction of hidden-service usage a relay should have observed. We use these fractions to remove implausible statistics in step 4. Then we extrapolate network totals in step 5, where each extrapolation is based on the report from a single relay. Finally, in step 6 we select daily averages from these network totals which constitutes our result.

[*]This material is based upon work supported by the National Science Foundation under Grant No. CNS-1111539. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

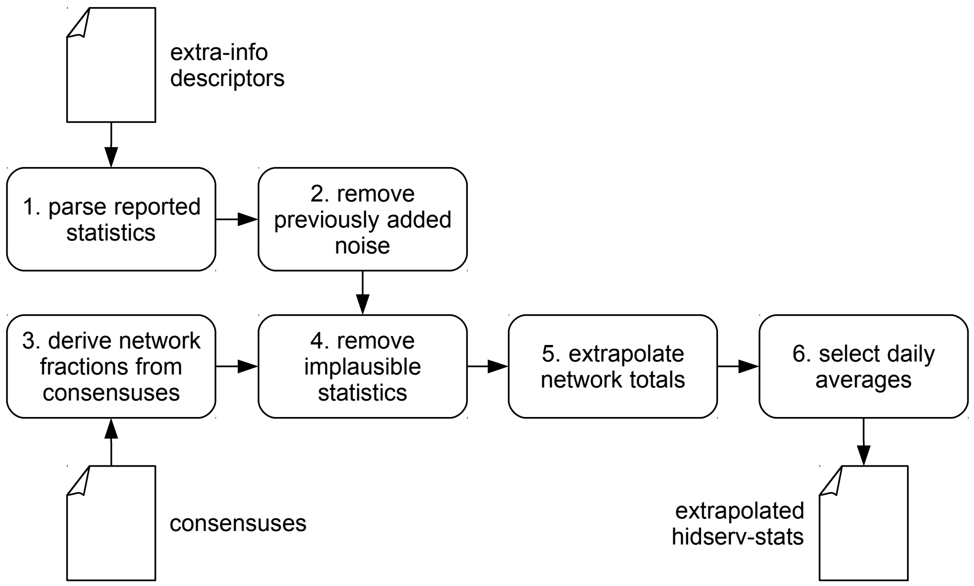

extra-info descriptors
1. parse reported statistics
2. remove previously added noise
3. derive network fractions from consensuses
4. remove implausible statistics
5. extrapolate network totals
6. select daily averages
consensuses
extrapolated hidserv-stats

Figure 1: Overview of the extrapolation method used for extrapolating network totals from hidden-service statistics.

1 Parsing reported statistics

There are two types of documents produced by Tor relays that we consider in our analysis. The first are extra-info descriptors that contain hidden-service statistics if a relay opts in to reporting them. The second are consensuses that indicate what fraction of hidden-service descriptors a hidden-service directory has observed and what fraction of rendezvous circuits a relay has handled.

We start by describing how we're parsing and processing hidden-service statistics from extra-info descriptors. Figure 2 shows the number of statistics reported by day, and Figure 3 shows a sample. The relevant parts for this analysis are:

- The `extra-info` line tells us which relay reported these statistics, which we need to know to derive what fraction of hidden-service activity this relay has observed.

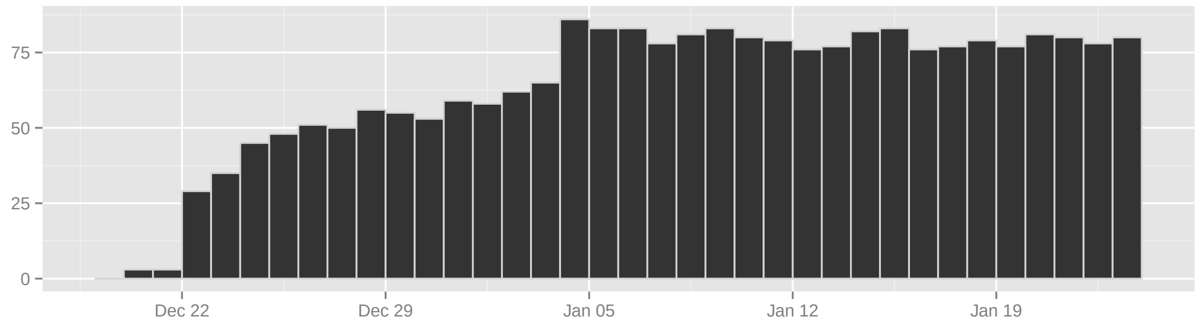


Figure 2: Number of reported hidden-service statistics.

```
extra-info ryroConoha F528DED21EACD2E4E9301EC0AABD370EDCAD2C47
[...]
hidserv-stats-end 2015-01-01 16:17:33 (86400 s)
hidserv-rend-relayed-cells 152599508 delta_f=2048 epsilon=0.30 bin_size=1024
hidserv-dir-onions-seen 91 delta_f=8 epsilon=0.30 bin_size=8
```

Figure 3: Sample hidden-service statistics contained in an extra-info descriptor.

- The `hidserv-stats-end` line tells us when the statistics interval ended, and, together with the interval length, when it started.
- The `hidserv-rend-relayed-cells` line tells us the number of cells that the relay handled on rendezvous circuits, and it tells us how this number has been obfuscated by the relay. The value for `bin_size` is the bin size used for rounding up the originally observed cell number, and the values for `delta_f` and `epsilon` are inputs for the additive noise following a Laplace distribution. For more information on how obfuscation is performed, please see Tor proposal 238.[1]
- And finally, the `hidserv-dir-onions-seen` line tells us the number of .onion addresses that the relay observed in published hidden-service descriptors in its role as hidden-service directory.

2 Removing previously added noise

When processing hidden-service statistics, we need to handle the fact that they have been obfuscated by relays. As first step, we're attempting to remove the additive Laplace-distributed noise by rounding up or down to the nearest multiple of `bin_size`. The idea is that it's most likely that noise was added to the closest right side of a bin than to the right side of another bin. In step two, we're subtracting half of `bin_size`, because the relay added between 0 and `bin_size`−1 to the originally observed value.

Following these steps, the statistics reported in Figure 3 are processed to 152599040 cells and 84 .onion addresses. For the subsequent analysis we're also converting cells/day to bits/second by multiplying cell numbers with 512 bytes/cell, multiplying with 8 bits/byte, and dividing by 86400 seconds/day.[2] As a result we obtain 7.2 Mbit/s in the given sample.

Figure 4 shows parsed values after removing previously added noise. Negative values are the result of relays adding negative Laplace-distributed noise values to very small observed values, which we cannot remove easily. We will describe an attempt to remove such values in Sections 4 and 6.

[1] https://gitweb.torproject.org/torspec.git/tree/proposals/238-hs-relay-stats.txt

[2] The originally published report had another quotient of 2 in this calculation, based on the false assumption that we would otherwise double-count cells going in incoming and outgoing direction. But this is not the case: we're counting each cell going in either incoming or outgoing direction only once when relaying it. All subsequent results have been fixed accordingly.

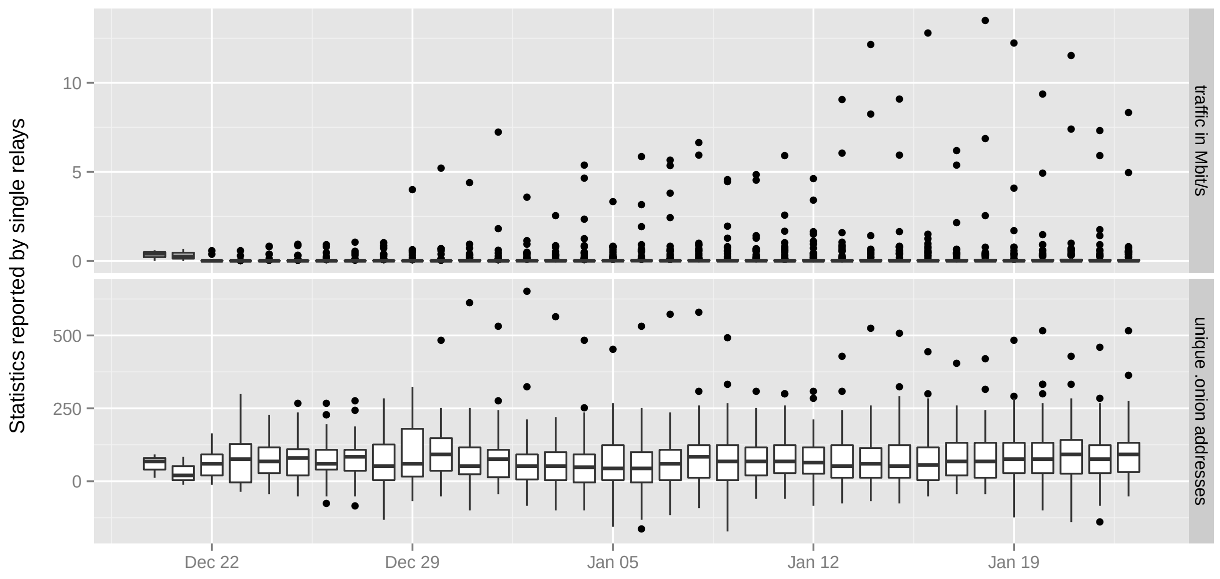


Figure 4: Hidden-service statistics reported by single relays after removing noise that was added by relays. Negative values are the result of relays adding negative Laplace-distributed noise values to very small observed values.

3 Deriving network fractions from consensuses

The second document type that we consider in our analysis are consensuses. The probability of choosing a relay as rendezvous point varies a lot between relays, and not all hidden-service directories handle the same number of hidden-service descriptors. Fortunately, we can derive what fraction of rendezvous circuits a relay has handled and what fraction of descriptors a directory was responsible for.

Figure 5 shows the consensus entry of the relay that submitted the sample hidden-service statistics mentioned above, plus neighboring consensus entries.

The first fraction that we compute is the probability of a relay to be selected as rendezvous point. Clients only select relays with the `Fast` flag and in some cases the `Stable` flag, and they weight relays differently based on their bandwidth and depending on whether they have the `Exit` and/or `Guard` flags. (Clients require relays to have the `Stable` flag if they attempt to establish a long-running connection, e.g., to a hidden SSH server, but in the following analysis, we assume that most clients establish connections that don't need to last for long, e.g., to hidden webservers.) Clients weight the bandwidth value contained in the consensus entry with the value of `Wmg`, `Wme`, `Wmd`, or `Wmm`, depending on whether the relay has only the `Guard` flag, only the `Exit` flag, both such flags, or neither of them.

Our sample relay, `ryroConoha`, has the `Fast` flag, a bandwidth value of 117000, and neither `Guard` nor `Exit` flag. Its probability for being selected as rendezvous point is calculated as $117000 \times 10000/10000$ divided by the sum of all such weights in the consensus, in this case 1.42%.

The second fraction that we can derive from this consensus entry is the fraction of descriptor

```
valid-after 2015-01-01 12:00:00
[...]
r adc 9PodlaV/b092ts4wpVLjq5d6mOs [...]
s Fast HSDir Running Stable V2Dir Valid
[...]
r AlongProxy 9QtBPk7CqqYP7WHNJJfgSVpJcyo [...]
s Fast HSDir Running Stable V2Dir Valid
[...]
r horizons3 9ScwmKcR+EXl0aJPnTj5O4ag8iA [...]
s Fast HSDir Running Stable V2Dir Valid
[...]
r ryroConoha 9Sje0h6s0uTpMB7Aqr03DtytLEc [...]
s Fast HSDir Running Stable V2Dir Valid
v Tor 0.2.6.1-alpha-dev
w Bandwidth=117000
p reject 1-65535
[...]
bandwidth-weights [...] Wmd=0 Wme=0 Wmg=3701 Wmm=10000
```

Figure 5: Sample consensus entries of relay `ryroConoha` that reports hidden-service statistics and of the three hidden-service directories preceding it.

space that this relay was responsible for in its role as hidden-service directory. The Tor Rendezvous Specification[3] contains the following definition that is relevant here:

> *A hidden service directory is deemed responsible for a descriptor ID if it has the HSDir flag and its identity digest is one of the first three identity digests of HSDir relays following the descriptor ID in a circular list.*

Based on the fraction of descriptor space that a directory was responsible for we can compute the fraction of descriptors that this directory has seen. Intuitively, one might think that these fractions are the same. However, this is not the case: each descriptor that is published to a directory is also published to two other directories. As a result we need to divide the fraction of descriptor space by *three* to obtain the fraction of descriptors observed the directory. Note that, without dividing by three, fractions of all directories would not add up to 100%.

In the sample consensus entry, we'd extract the base64-encoded fingerprint of the statistics-reporting relay, `9Sje0h6...`, and the fingerprint of the hidden-service directory that precedes the relay by three positions, `9PodlaV...`, and compute what fraction of descriptor space that is, in this case 0.071%. So, the relay has observed 0.024% of descriptors in the network.

Figure 6 shows calculated fractions of hidden-service activity observed by relays that report hidden-service statistics. The probability for being selected as rendezvous point is very small for most relays, with only very few relays having a realistic chance of being selected. In comparison, most relays have roughly the same (small) probability for observing a hidden-service descriptor with only few exceptions.

[3] https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt

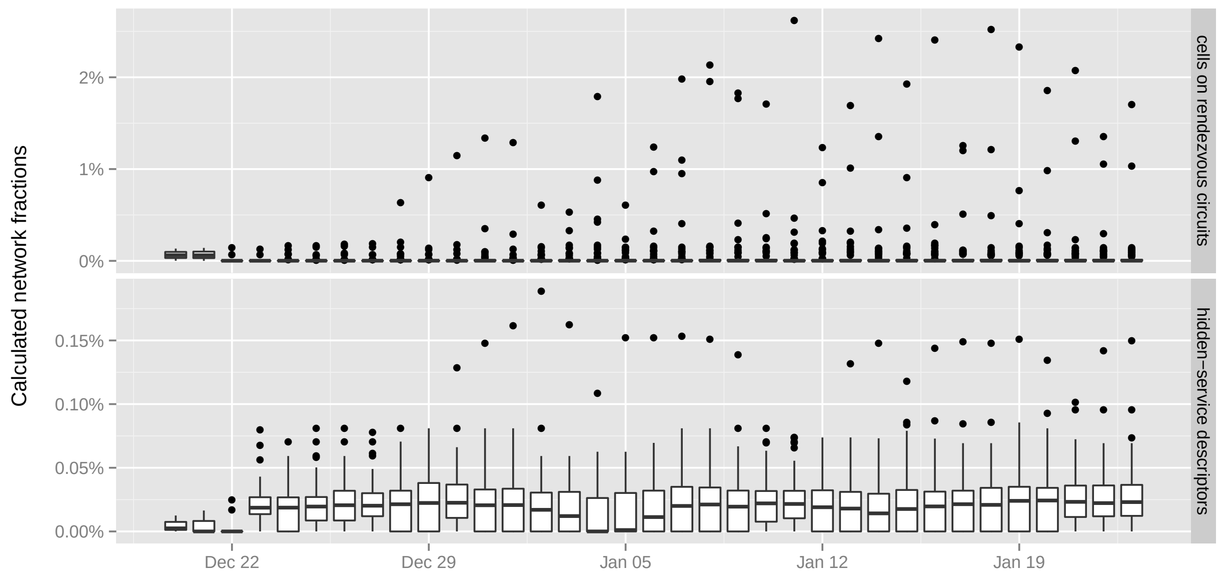


Figure 6: Calculated network fractions of relays observing hidden-service activity.

4 Removing implausible statistics

A relay that opts in to gathering hidden-service statistics reports them even if it couldn't plausibly have observed them. In particular, a relay with the `Exit` flag could not have been selected as rendezvous point as long as `Wmd` and `Wme` are zero, and a relay that did not have the `HSDir` flag could not have observed a single .onion address.

Figure 7 shows distributions of reported statistics of relays with calculated fractions of exactly zero. These reported values approximately follow the plotted Laplace distributions with $\mu = 0$ and $b = 2048/0.3$ or $b = 8/0.3$ as defined for the respective statistics, which gives us confidence that the vast majority of these reported values are just noise. In the following analysis, we exclude relays with calculated fractions of exactly 0.

Another kind of implausible statistics are very high or very low absolute reported numbers. These numbers could be the result of adding very large positive or negative numbers from the Laplace distribution. In theory, a single relay, with non-zero probability of observing hidden-service activity, could have added noise from $-\infty$ to ∞. Further, relays could lie about hidden-service usage and report very low or very high absolute values in their statistics in an attempt to derail statistics. It seems difficult to define a range of plausible values, and such a range might change over time. It seems easier to handle these extreme values by treating a certain fraction of extrapolated statistics as outliers, which is what we're going to do in Section 6.

5 Extrapolating network totals

We are now ready to extrapolate network totals from reported statistics. We do this by dividing reported statistics by the calculated fraction of observations made by the reporting relay. The

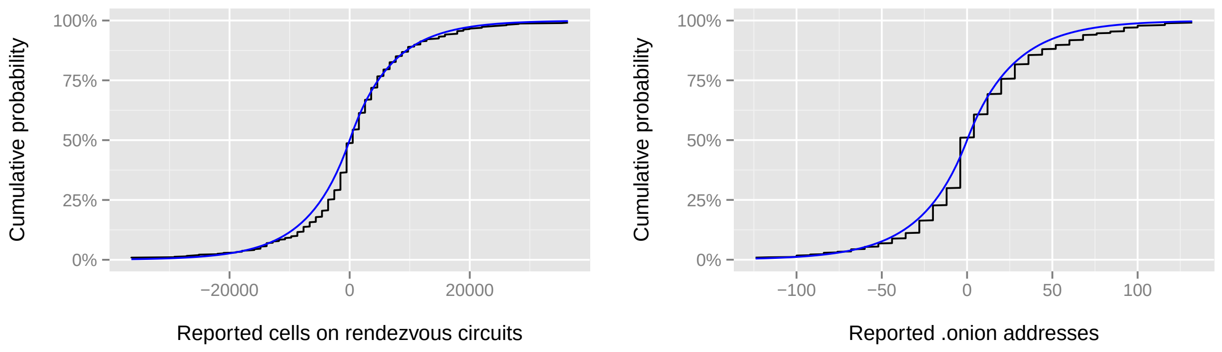


Figure 7: Statistics reported by relays with calculated probabilities of observing these statistics of zero. The blue lines show Laplace distributions with $\mu = 0$ and $b = 2048/0.3$ or $b = 8/0.3$ as defined for the respective statistics. The lowest 1% and highest 1% of values have been removed for display purposes.

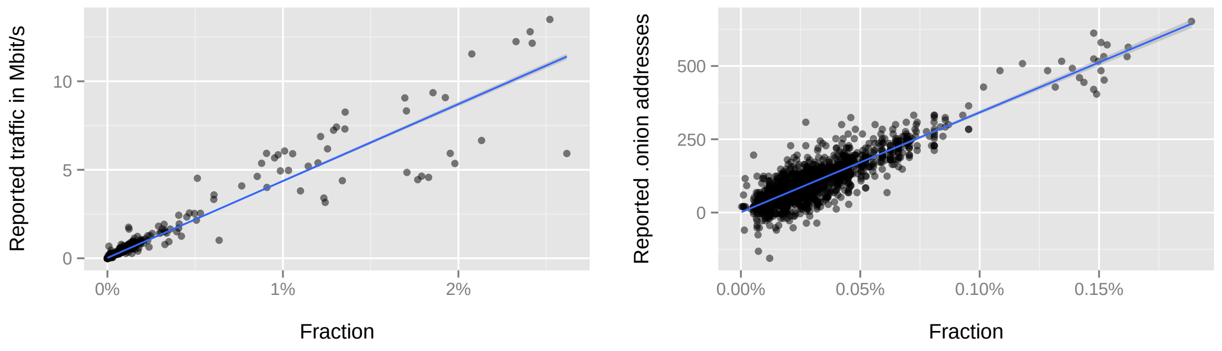


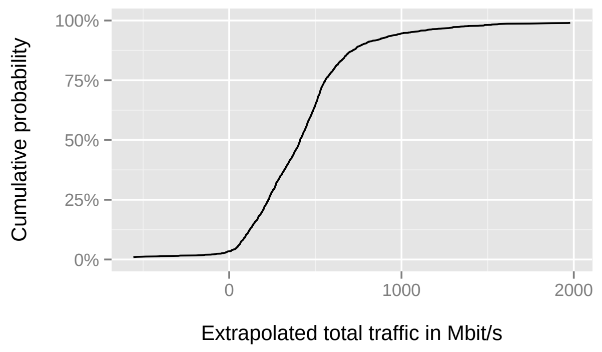
Figure 8: Correlation between reported hidden-service activity and calculated probability for observing such activity.

underlying assumption is that statistics grow linearly with calculated fractions. Figure 8 shows that this is roughly the case.

While we can expect this method to work as described for extrapolating cells on rendezvous circuits, we need to take another step for estimating the number of unique .onion addresses in the network. The reason is that a .onion address is not only known to a single relay, but to a couple of relays, all of which include that .onion address in their statistics. We need to subtract out the multiple counting of .onion addresses to come up with a network-wide number of unique .onion addresses.

As an approximation, we assume that a hidden service publishes its descriptor to *twelve* directories over a 24-hour period: the service stores *two* replicas per descriptor using different descriptor identifiers, both descriptor replicas get stored to *three* different hidden-service directories each, and the service changes descriptor identifiers once every 24 hours which leads to *two* different descriptor identifiers per replica.

To be clear, this approximation is not entirely accurate. For example, the two replicas or the descriptors with changed descriptor identifiers could have been stored to the same directory. As another example, hidden service directories might have joined or left the network and other



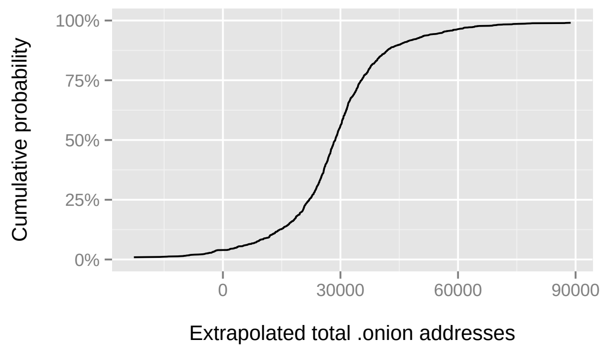


Figure 9: Distribution of extrapolated network totals for all days in the analysis period, excluding lowest 1% and highest 1% for display purposes.

directories might have become responsible for storing a descriptor which also include that .onion address in their statistics. However, for the subsequent analysis, we assume that neither of these cases affects results substantially.

We can now extrapolate reported unique .onion addresses to network totals. Figure 9 shows the distributions of extrapolated network totals for all days in the analysis period.

6 Selecting daily averages

As last step in the analysis, we aggregate extrapolated network totals for a given day to obtain a daily average. We considered a few options for calculating the average, each of which having their advantages and drawbacks.

We started looking at the *weighted mean* of extrapolated network totals, which is the mean of all values but which uses relay fractions as weights, so that smaller relays cannot influence the overall result too much. This metric is equivalent to summing up all reported statistics and dividing by the sum of network fractions of reporting relays. The nice property of this metric is that it considers all statistics reported by relays on a given day. But this property is also the biggest disadvantage: single extreme statistics can affect the overall result. For example, relays that added very large noise values to their statistics cannot be filtered out. The same holds for relays that lie about their statistics.

Another metric we looked at was the *weighted median*, which also takes into account that relays contribute different fractions to the overall statistic. While this metric is not affected by outliers, basing the daily statistics on the data from a single relay doesn't seem very robust.

In the end we decided to pick the *weighted interquartile mean* as metric for the daily average. For this metric we order extrapolated network totals by their value, discard the lower and the upper quartile by weight, and compute the weighted mean of the remaining values. This metric is robust against noisy statistics and lying relays and considers half of the reported statistics.

We further define a threshold of 1% for the total fraction of relays reporting statistics. If less than these 1% of relays report statistics on a given day, we don't display that day in the end results. Figure 10 shows total calculated network fractions per day, and Figure 11 shows weighted interquartile of the extrapolated network totals per day.

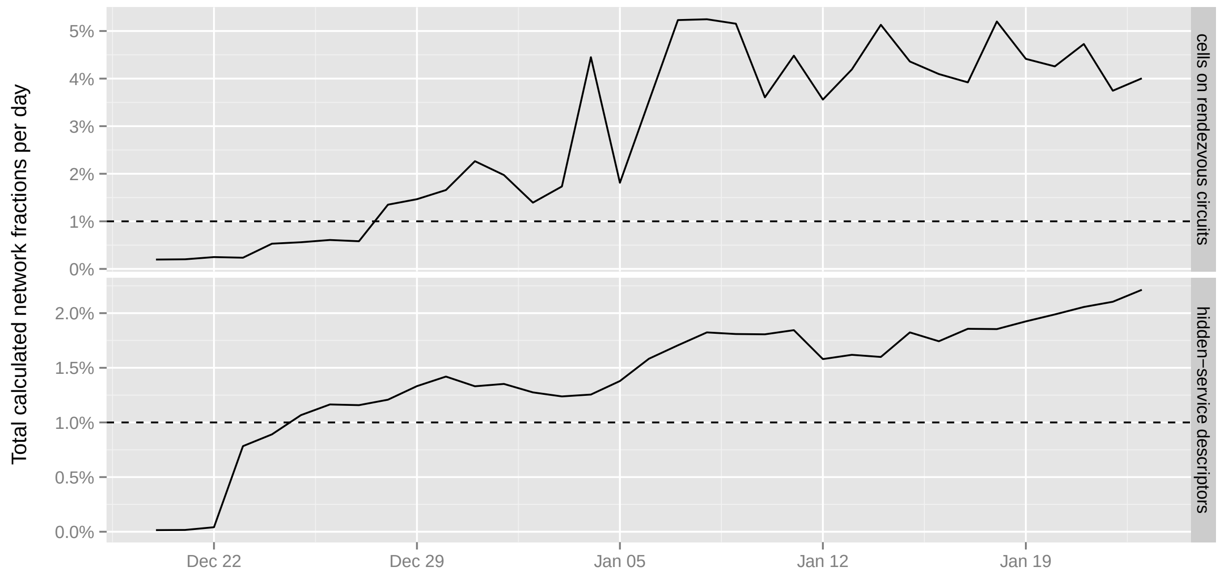


Figure 10: Total calculated network fractions per day.

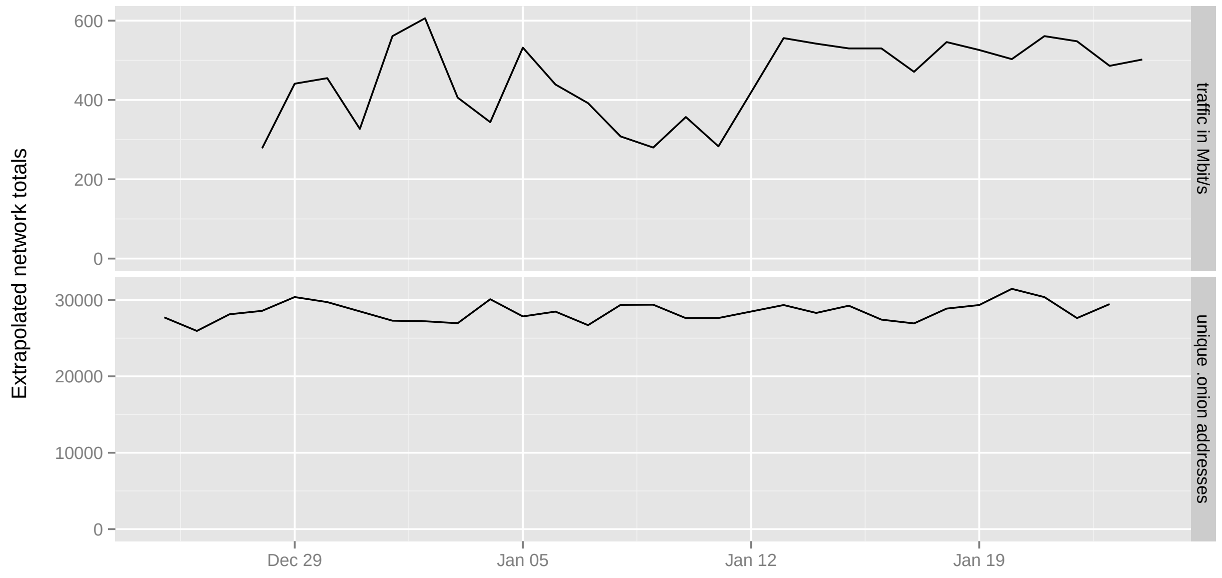


Figure 11: Daily averages of extrapolated network totals, calculated as weighted interquartile means of extrapolations based on statistics by single relays.

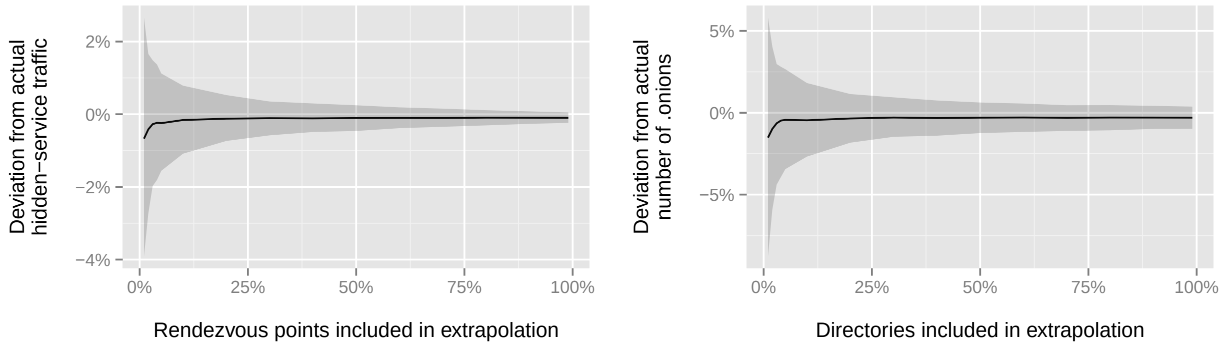


Figure 12: Median and range from 2.5th to 97.5th percentile of simulated extrapolations.

Evaluation

We conducted two simulations to demonstrate that the extrapolation method used here delivers approximately correct results and to gain some sense of confidence in the results if only very few relays report statistics.

In the first simulation we created a network of 3000 middle relays with consensus weights following an exponential distribution. We then randomly selected relays as rendezvous points and assigned them, in total, 10^9 cells containing hidden-service traffic in chunks with chunk sizes following an exponential distribution with $\lambda = 0.0001$. Each relay obfuscated its observed cell count and reported obfuscated statistics. Finally, we picked different fractions of reported statistics and extrapolated total cell counts in the network based on these. We also conducted a second simulation with 3000 hidden-service directories and 40000 hidden services, each of them publishing descriptors to 12 directories.

Figure 12 shows the median and the range between 2.5th and 97.5th percentile for the extrapolation. As long as we included at least 1% of relays by consensus weight in the extrapolation, network totals did not deviate by more than 5% in positive or negative direction.

Conclusion

In this report we described a method for extrapolating network totals from the two recently added hidden-service statistics. We showed that we can extrapolate network totals with reasonable accuracy as long as at least 1% of relays report these statistics.

Acknowledgements

Thanks to Aaron Johnson for providing invaluable feedback on extrapolating statistics and on running simulations. Thanks to the relay operators who enabled the new hidden-service statistics on their relays and provided us with the data to write this report.