An Analysis of Tor Bridge Stability
— Making BridgeDB give out at least one stable bridge per user —

Karsten Loesing
karsten@torproject.org

Tor Tech Report 2011-10-001
October 31, 2011

1 Introducing the unstable bridges problem

As of October 2011, the Tor network consists of a few hundred thousand clients, 2 400 public relays, and about 600 non-public bridge relays. Bridge relays (in the following: bridges) are entry points which are not publicly listed to prevent censors from blocking access to the Tor network. Censored users request a small number of typically three bridge addresses from the BridgeDB service via email or http and then connect to the Tor network only via these bridges. If all bridges that a user knows about suddenly stop working, the user needs to request a new set of bridge addresses from BridgeDB. However, BridgeDB memorizes the user's email or IP address and only gives out new bridges every 24 hours to slow down enumeration attempts. The result is that a user who is unlucky enough to receive only unreliable bridges from BridgeDB won't be able to connect to the Tor network for up to 24 hours before requesting a new set of bridges.

In this report we propose that BridgeDB keeps bridge stability records, similar to how the directory authorities keep relay stability records, and includes at least one stable bridge in its responses to users. In fact, BridgeDB currently attempts to do this by including at least one bridge with the Stable flag assigned by the bridge authority in its results. This approach is broken for two reasons: The first reason is that the algorithm that the bridge authority uses to assign the Stable flag is broken to the extent that almost every bridge has the Stable flag assigned. The second reason is that the Stable flag was designed for clients to select relays for long-running streams, not for BridgeDB to select reliable entry points into the Tor network. A better metric for stable bridges would be based on bridge uptime and on the frequency of IP address changes. We propose such a metric and evaluate its effectiveness for selecting stable bridges based on archived bridge directories.

2 Defining a new bridge stability metric

The directory authorities implement a few relay stability metrics to decide which of the relays to assign the Guard and Stable flag [1, 2]. The requirements for stable bridges that we propose here are similar to the entry guard requirements. That is, stable bridges should have a higher fractional uptime than non-stable ones. Further, a stable bridge should be available under the same IP address and TCP port. Otherwise, bridge users who only know a bridge address won't be able to connect to the bridge once it changes its address or port. We propose the following requirements for a bridge to be considered stable in the style of the Guard and Stable flag definition:

> A bridge is considered stable if its *Weighted Mean Time Between Address Change* is at least the median for known active bridges or at least 30 days, if it is 'familiar', and if its *Weighted Fractional Uptime* is at least the median for 'familiar' active bridges or at least 98 %. A bridge is 'familiar' if 1/8 of all active bridges have appeared more recently than it, or if it has been around for a *Weighted Time* of 8 days.

This bridge stability definition contains three main requirements:

- The *Weighted Mean Time Between Address Change (WMTBAC)* metric is used to track the time that a bridge typically uses the same IP address and TCP port. The (unweighted) MTBAC measures the average time between last using address and port a_0 to last using address and port a_1. This metric is weighted to put more emphasis on recent events than on past events. Therefore, past address sessions are discounted by factor 0.95 every 12 hours. The current session is not discounted, so that a WMTBAC value of 30 days can be reached after 30 days at the earliest.
- The *Weighted Fractional Uptime (WFU)* metric measures the fraction of bridge uptime in the past. Similar to WMTBAC, WFU values are discounted by factor 0.95 every 12 hours, but in this case including the current uptime session.
- The *Weighted Time (WT)* metric is used to calculate a bridge's WFU and to decide whether a bridge is around long enough to be considered 'familiar.' WT is discounted similar to WMTBAC and WFU, so that a WT of 8 days can be reached after around 16 days at the earliest.

All three requirements consist of a dynamic part that depends on the stability of other bridges (e.g., "A bridge is familiar if 1/8 of all active bridges have appeared more recently than it, . . . ") and a static part that is independent of other bridges (e.g., ". . . or if it has been around for a Weighted Time of 8 days."). The dynamic parts ensure that a certain fraction of bridges is considered stable even in a rather unstable network. The static parts ensures that rather stable bridges are not excluded even when most other bridges in the network are stable.

3 Extending BridgeDB to track bridge stability

There are at least two code bases that could be extended to track bridge stability and include at least one stable bridge in BridgeDB results: the bridge authority and BridgeDB. The decision for extending either code base affects the available data for tracking bridge stability and is therefore discussed here.

The bridge authority maintains a list of all active bridges. Bridges register at the bridge authority when joining the network, and the bridge authority periodically performs reachability tests to confirm that a bridge is still active. The bridge authority takes snapshots of the list of active bridges every 30 minutes and copies these snapshots to BridgeDB. BridgeDB parses these half-hourly snapshots and gives out bridges to users based on the most recently known snapshot.

The bridge stability history can be implemented either in the bridge authority code or in BridgeDB. On the one hand, an implementation in BridgeDB has the disadvantage that bridge reachability data has a resolution of 30 minutes whereas the bridge authority would learn about bridges joining or leaving the network immediately. On the other hand, the bridge stability information is not used by anything in the Tor software, but only by BridgeDB. Implementing this feature in BridgeDB makes more sense from a software architecture point of view. In the following we assume that BridgeDB will track bridge stability based on half-hourly snapshots of active bridge lists, the bridge network statuses.

4 Simulating bridge stability using archived data

We can analyze how BridgeDB would track bridge stability and give out stable bridges by using archived bridge descriptors. These archives contain the same descriptors that BridgeDB uses, but they are public and don't contain any IP addresses or sensitive pieces of information. In Section 4.1 we look at the problem of missing data due to either the bridge authority or BridgeDB failing and at the effect on tracking bridge stability. We then touch the topic of how bridge descriptors are sanitized and how we can glue them back together for our analysis in Section 4.2. Next, we examine typical bridge stability values as requirements for considering a bridge as stable in Section 4.3. In Section 4.4 we estimate what fraction of bridges would be considered as stable depending on the chosen stability requirements. Finally, in Section 4.5 we evaluate how effective different requirement combinations are for selecting stable bridges. Result metrics are how soon selected bridges change their address or what fractional uptime selected bridges have in the future.

4.1 Handling missing bridge status data

The bridge status data that we use in this analysis and that would also be used by BridgeDB to track bridge stability is generated by the bridge authority and copied over to BridgeDB every 30 minutes. Figure 1 shows the number of running bridges contained in these snapshots from July 2010 to June 2011.

For most of the time the number of bridges is relatively stable. But there are at least two irregularities, one in July 2010 and another one in February 2011, resulting from problems with the bridge authority or the data transfer to the BridgeDB host. Figure 2 shows these two intervals in more detail.

The missing data from July 14 to 27, 2010 comes from BridgeDB host not accepting new descriptors from the bridge authority because of an operating system upgrade of the BridgeDB

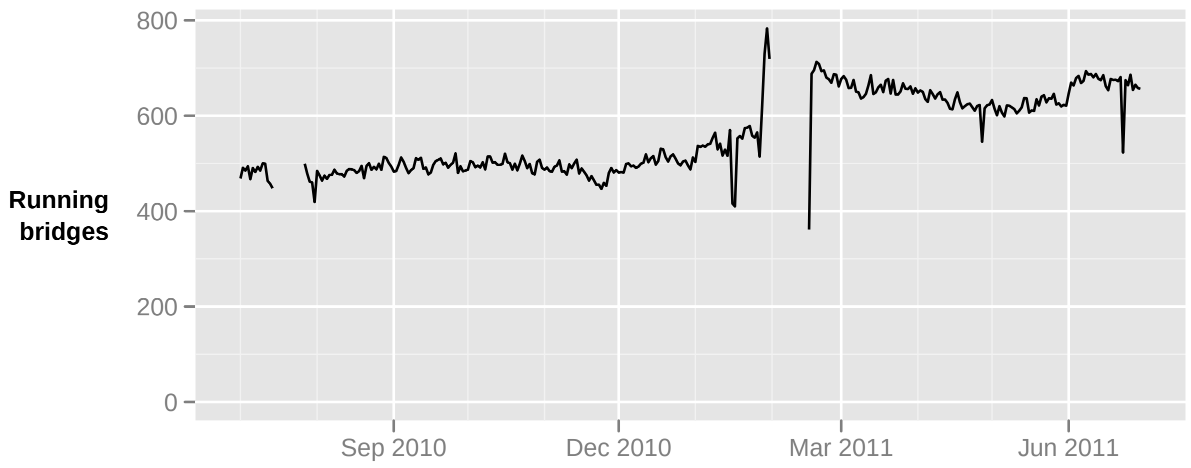

Figure 1: Median number of running bridges as reported by the bridge authority

host. During this time, the bridge authority continued to work, but BridgeDB was unable to learn about new bridge descriptors from it.

During the time from January 27 to February 16, 2011, the `tor` process running the bridge authority silently died twice after a Tor version upgrade, but the script to copy descriptors to BridgeDB kept running. In this case, BridgeDB received fresh tarballs containing stale descriptors with a constant number of 687 relays, visualized in light gray. These stale descriptors have been excluded from the sanitized descriptors and the subsequent analysis. The bridge authority was restarted on February 16, 2011, resulting in the number of running bridges slowly stabilizing throughout the day.

Both this analysis and a later implementation in BridgeDB need to take extended phases of missing or stale data into account.

4.2 Detecting address changes in sanitized descriptors

The bridge descriptor archives that we use in this analysis have been sanitized to remove all addresses and otherwise sensitive parts [3]. Part of this sanitizing process is that bridge IP addresses are replaced with keyed hashes using a fresh key every month. More precisely, every bridge IP address is replaced with the private IP address `10.x.x.x` with `x.x.x` being the 3 most significant bytes of `SHA-256(IP address | bridge identity | secret)`.

A side-effect of this sanitizing step is that a bridge's sanitized IP address changes at least once per month, even if the bridge's real IP address stays the same. We need to detect these artificial address changes and distinguish them from real IP address changes.

In this analysis we use a simple heuristic to distinguish between real IP address changes and artifacts from the sanitizing process: Whenever we find that a bridge has changed its IP address from one month to the next, we look up how long both IP addresses were in use in either month. If both addresses were contained in bridge descriptors that were published at least 36 hours apart, we consider them stable IP addresses and attribute the apparent IP address

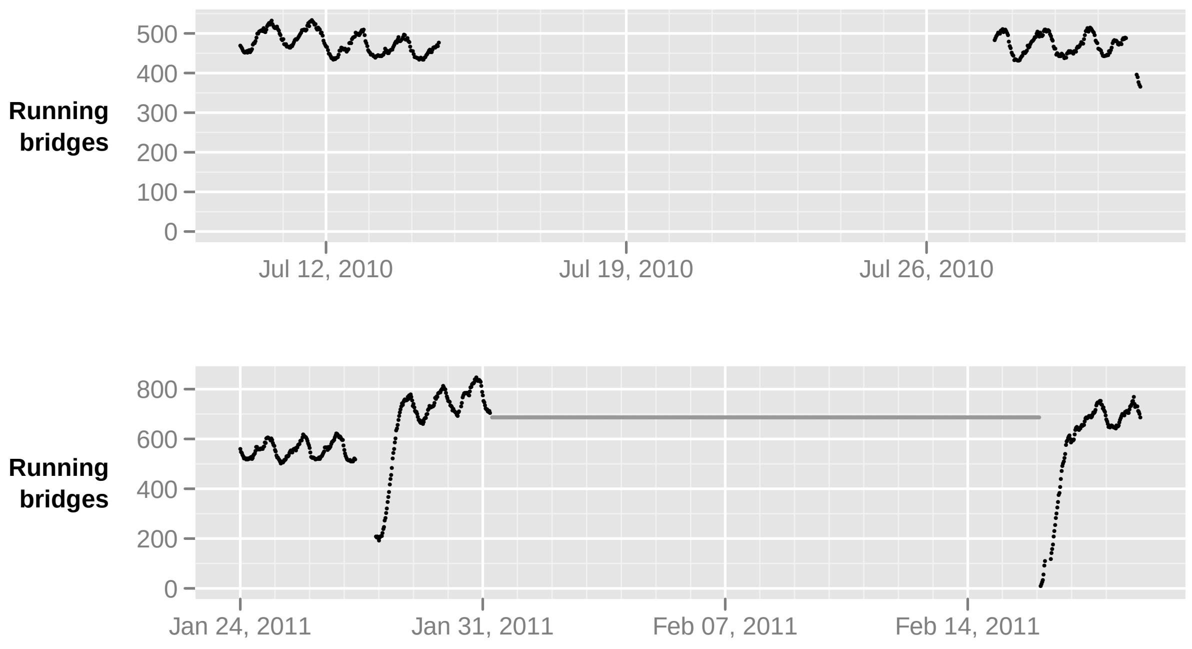

Figure 2: Number of Running bridges during phases when either the bridge authority or the BridgeDB host were broken

change to the sanitizing process. Otherwise, we assume the bridge has really changed its IP address. Obviously, this simple heuristic might lead us to false conclusions in some cases. But it helps us handle cases when bridges rarely or never change their IP address which would otherwise suffer from monthly address changes in this analysis.

4.3 Examining typical stability metric values

The definition of bridge stability on page 2 contains three different metrics, each of which having a dynamic and a static part. The dynamic parts compares the value of a bridge's stability metric to the whole set of running bridges. Only those bridges are considered as stable that exceed the median value (or the 12.5th percentile) of all running bridges. The static requirement parts are fixed values for all stability metrics that don't rely on the stability of other bridges.

Figure 3 visualizes the dynamic (solid lines) and static parts (dashed lines) of all three requirements. The dynamic WMTBAC requirements are higher than previously expected. A value of 60 means that, on average, bridges keep their IP address and port for 60 days. The dynamic values are cut off at 30 days by the static requirement which should be a high enough value. The goal here is to give blocked users a stable enough set of bridges so that they don't have to wait another 24 hours before receiving new ones.

We can further see that the dynamic requirements are relatively stable over time except for the two phases of missing bridge status data. The first phase in July 2010 mostly affects WT, but neither WMTBAC nor WFU. The second phase in February 2011 affects all three metrics.

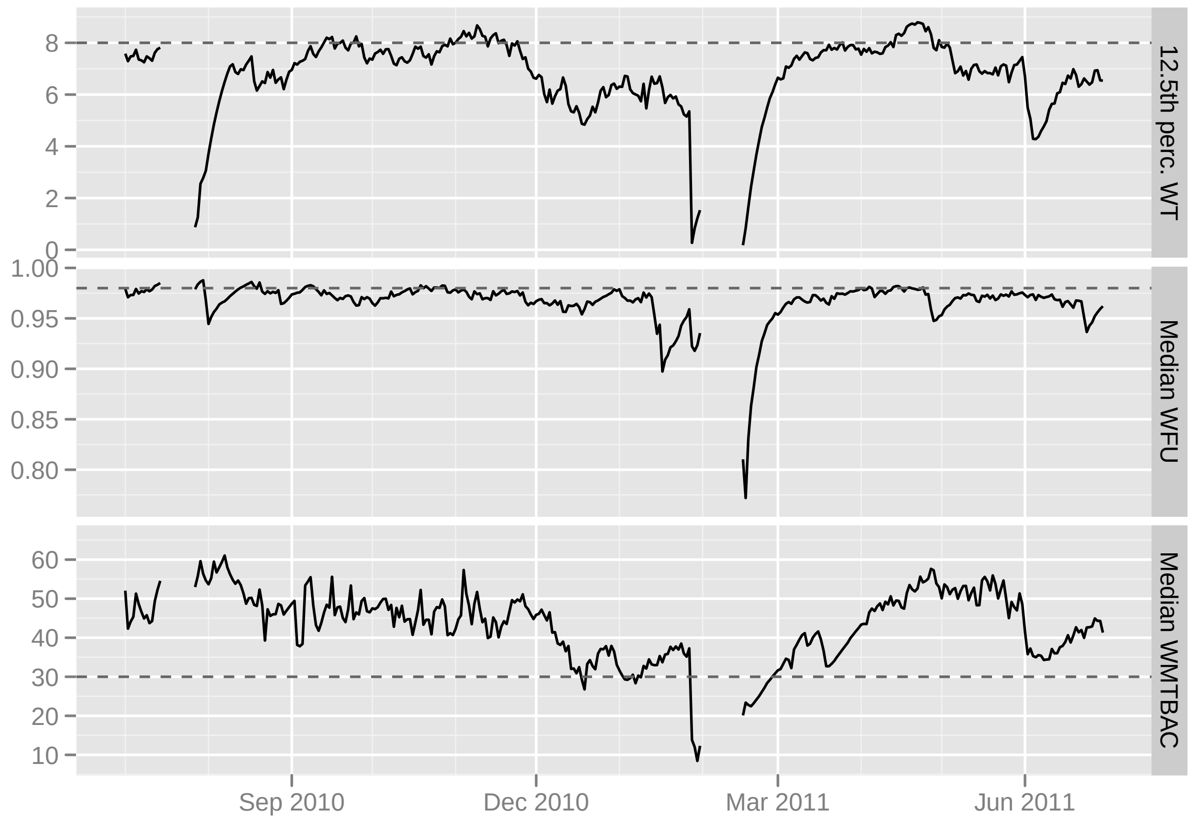

Figure 3: Dynamic requirements for considering a bridge as stable

We can expect the selection of stable bridges during February 2010 to be more random than at other times.

4.4 Estimating fractions of bridges considered as stable

Requiring a bridge to meet or exceed either or both WMTBF or WFU metric results in considering only a subset of all bridges as stable. The first result of this analysis is to outline what fraction of bridges would be considered as stable if BridgeDB used either or both requirements. In theory, all parameters in the bridge stability definition on page 2 could be adjusted to change the set of stable bridges or focus more on address changes or on fractional uptime. We're leaving the fine-tuning for future work when specifying and implementing the BridgeDB extension.

Figure 4 shows the fraction of stable bridges over time. If we only require bridges to meet or exceed the median WMTBAC or the fixed value of 30 days, roughly 55 % of the bridges are considered as stable. If bridges are only required to meet or exceed the WT and WFU values, about $7/8 \times 1/2 = 43.75$ % of bridges are considered as stable. Requiring both WFU and WMTBAC leads to a fraction of roughly 35 % stable bridges.

The fraction of 33 % stable bridges seems appropriate if 1 out of 3 bridges in the BridgeDB results is supposed to be a stable bridge. If more than 1 bridge should be a stable bridge, the

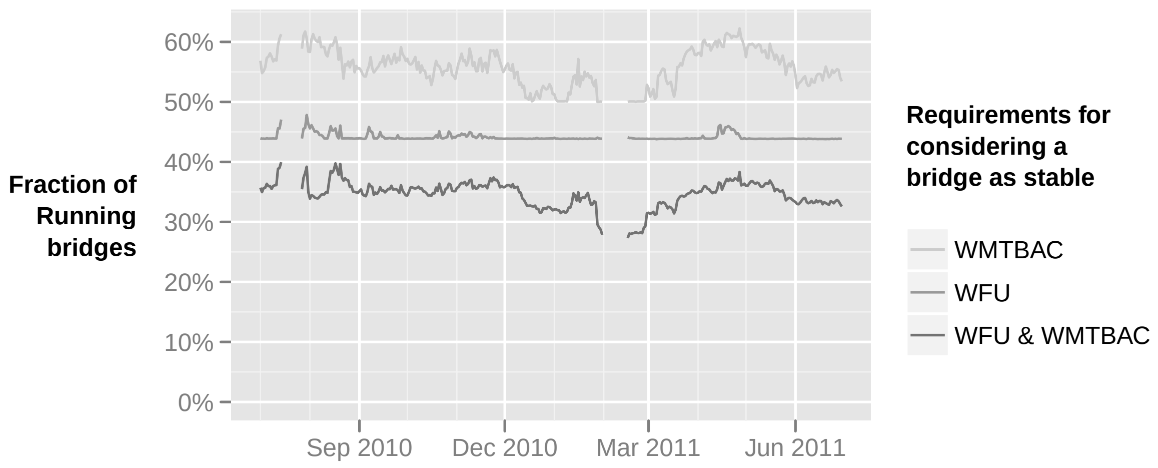

Figure 4: Impact of requiring stable bridges to meet or exceed the median WFU and/or WMTBAC on the fraction of running bridges considered as stable

requirements need to be lowered, so that a higher fraction of bridges is considered stable. Otherwise, the load on stable bridges might become too high.

4.5 Evaluating different requirements on stable bridges

The main purpose of this analysis is to compare the quality of certain requirements and requirement combinations on the stability of selected bridges. Similar to the previous section, we only compare whether or not the WMTBAC or WFU requirement is used, but don't change their parameters.

The first result is the future uptime that we can expect from a bridge that we consider stable. We calculate future uptime similar to past uptime by weighting events in the near future more than those happening later. We are particularly interested in the almost worst-case scenario here, which is why we're looking at the 10th percentile weighted fractional uptime in the future. This number means that 10 % of bridges have a weighted fractional uptime at most this high and 90 % of bridges have a value at least this high.

Figure 5 visualizes the four possible combinations of using or not using the WMTBAC and WFU requirements. In this plot, the "WFU & WMTBAC" and "WFU" lines almost entirely overlap, meaning that the WMTBAC requirement doesn't add anything to future uptime of selected bridges. If the WFU requirement is not used, requiring bridges to meet the WMTBAC requirement increases future uptime from roughly 35 % to maybe 55 %. That means that there is a slight correlation between the two metrics, which is plausible.

The second result is the time that a selected bridge stays on the same address and port. We simply measure the time that the bridge will keep using its current address in days. Again, we look at the 10th percentile. 90 % of selected bridges keep their address longer than this time.

Figure 6 shows for how long bridges keep their address and port. Bridges meeting both WFU and WTMBAC requirements keep their address for 2 to 5 weeks. This value decreases to 1

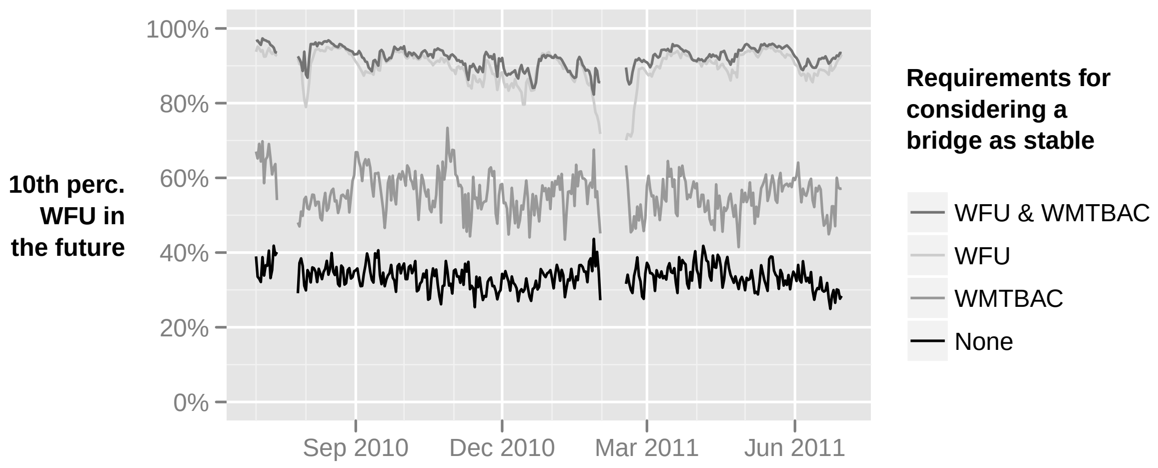

Figure 5: Impact of requiring stable bridges to meet or exceed the median WFU and/or WMTBAC on the 10th percentile weighted fractional uptime in the future

to 3 weeks when taking away the WFU requirement, which is also a result of the two metrics beeing correlated. The bridges that only meet the WFU requirement and not the WMTBAC requirement change their address within the first week. If we don't use any requirement at all, which is what BridgeDB does today, 10 % of all bridges change their address within a single day.

5 Concluding the bridge stability analysis

In this report we propose to extend BridgeDB to make it give out at least one stable bridge per user. Bridge stability can be calculated based on bridge status information over time, similar to how the directory authorities calculate relay stability. The bridge stability metric proposed here is based on a bridge's past uptime and the frequency of changing its address and/or port. Requiring at least 1 bridge of the 3 to be given out to users greatly reduces the worst case probability of all bridges being offline or changing their addresses or ports. The price for this increase in stability is that stable bridges will be given out more often than non-stable bridges and will therefore see more usage.

We suggest to implement the described bridge stability metric in BridgeDB and make it configurable to tweak the requirement parameters if needed. Maybe it turns out to be more useful to lower the requirements for a bridge to become stable and give out two stable bridges per response. It's also possible that the requirement for a bridge to keep its address becomes less important in the future when bridge clients can request a bridge's current address from the bridge authority. All these scenarios can be analyzed before deploying them using archived data as done in this report.

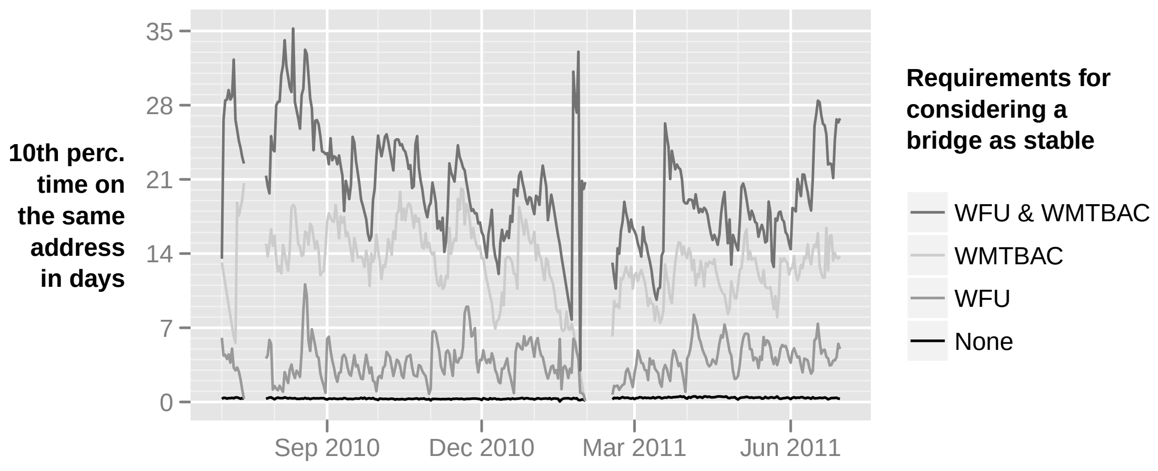

Figure 6: Impact of requiring stable bridges to meet or exceed the median WFU and/or WMTBAC on the 10th percentile time on the same address

References

[1] Roger Dingledine and Nick Mathewson. Tor directory protocol, version 3. `https://gitweb.torproject.org/tor.git/blob_plain/HEAD:/doc/spec/dir-spec.txt`.

[2] Karsten Loesing. An analysis of Tor relay stability. Technical Report 2011-06-001, The Tor Project, June 2011.

[3] Karsten Loesing. Overview of statistical data in the Tor network. Technical Report 2011-03-001, The Tor Project, March 2011.