The Game Has Changed: Revisiting proxy distribution and game theory

Hassan Fares
University of Minnesota
Minneapolis, MN, USA
fares011@umn.edu

Omkar Fulsundar
University of Minnesota
Minneapolis, MN, USA
fulsu001@umn.edu

Nicholas Hopper
University of Minnesota
Minneapolis, MN, USA
hoppernj@umn.edu

Abstract

In 2019, Nasr *et al.* introduced a game-theoretic framework for evaluating censorship-resilient proxy distribution schemes. In light of recent trends in proxy-based circumvention tools, and developments and revelations about Internet censor capabilities, we revisit this framework. Specifically, we implement simulations that model ephemeral, browser- or mobile-based proxies with NAT restrictions as used in Lantern, Psiphon and Snowflake, and also model censors that use traffic analysis to inform proxy enumeration schemes. We show that "optimal" proxy distribution strategies that do not incorporate these advances are far from optimal, while the simple strategies used by ephmeral tools perform very well. Our results suggest there is a need for future research to better model the objective functions of proxy distribution strategies and censors.

Keywords

Proxy Distribution, Game Theory, Censorship, Simulation

1 Introduction

Freedom of speech and freedom of assembly are foundational principles for modern democratic society, and the Internet has become one of the most common and effective means of conveying expression that is likely to be suppressed and finding others who share such views. Because of this, in many regions of the world a variety of means are now used to block access to some information and systems available over the Internet. Circumvention tools, such as Tor [8], Psiphon [14] or Virtual Private Network (VPN) providers may help to circumvent such blocking through encryption (to hide content) and redirection (using proxies to avoid blocked addresses). However, (Internet) censors can also use these tools to discover and block the Internet addresses of proxies used by the tools.

Proxy Distribution. A Key problem facing circumvention tools is developing strategies to distribute proxy addresses, because of the conflict between allowing users to find proxies, and preventing censors from finding and blocking them. A variety of rate-limiting schemes have been used by deployed tools and suggested in the literature, which we discuss in Section 2. However, these schemes are typically heuristic in nature and it can be difficult to compare them with each other or decide whether they provide the best performance in this tradeoff.

Free and Open Communications on the Internet 2026(1), 14–22

Enemy at the Gateways. To address this problem, Nasr *et al.* [17] introduced a game-theoretic framework to model the proxy distribution problem. In their model, a global censor controls several clients and decides which proxies to block, and a *distributor* decides which proxies to distribute to each client. Given utility functions for both the censor and the distributor, they derive an optimal strategy for both at each stage in the game. Through simulations of several different utility functions and network settings, they show that their optimal distributor outperforms previous heuristics, even against an optimal censor.

Changes in Censorship and Circumvention. One limitation of the Nasr *et al.* framework is that it does not model some of the capabilities and differences between Internet censors that measurement studies have identified. As we discuss further in Section 2, Internet censors in different regions have different limitations based on the architecture of their censorship infrastructure, so it may not be the case that an optimal distributor for one censor is optimal for a different censor, or when considering users from multiple regions. Additionally, the framework does not consider "traffic analysis" attacks that identify proxies by traffic fingerprint or connection patterns, which are documented features of censorship strategies.

Moreover a recent trend in circumvention tools is the use of proxies that are either ephemeral or can rapidly transition between multiple IP addresses. The choice to change addresses or transition quickly to a new proxy is a strategic possibility that is not explored in [17]. This option may also influence the strategy of a censor with traffic analysis or fine-grained blocking capabilities, thus altering the utility function of both the distributor and the censor.

Research Questions. Given these changes, we decided to update the simulation scenarios of Nasr *et al.* to investigate the following research questions:
RQ1: How do the optimal censors in the framework perform against ephemeral proxy distribution schemes?
RQ2: How do prior proxy distribution schemes perform when censors implement traffic analysis strategies that can have false positive detection rates? What level of collateral damage will be inflicted by a censor in these cases?
RQ3: Given the differences in censorship infrastructures and realizable strategies across countries, how do proxy distribution schemes perform when we more realistically model multiple censors?

The rest of this paper is organized as follows. Section 2 discusses related work on proxy distribution and gives more details on the Nasr *et al.* framework. In Section 3, we describe how we update their simulation framework to model the changes required to answer our research questions RQ1–RQ3. We then discuss the results of our simulations in Sections 4–6, before discussing conclusions from our work and possible future work in Section 7.

2 Background and Related Work

2.1 Internet censorship

In network censorship, the censor identifies either specific sites or keywords that users should be unable to access. The censor deploys appliances that monitor network activity for indications that this content is being accessed. For instance, unencrypted content (either in request fields or responses) can be directly matched against keywords or URLs [4]; unencrypted domain name requests can be matched against site names [23]; source and destination IP addresses can be matched against known server addresses, and in the case of TLS and QUIC, Server Name Indication (SNI) fields can be matched against blocked site names. Once these indicators are detected, the appliance will block the connection, for example, by responding to DNS requests with an incorrect address record [23], sending TCP reset packets to both ends of a connection [3], or null-routing all connection attempts to specific IP addresses.

Circumvention tools evade this blocking through the use of a relay host or network, also called a proxy. Clients trying to access censored content frequently perform a rendezvous or discovery step to learn the network address of, and establish a connection to, a proxy. The client then connects with the relay, using an encrypted transport protocol, and its content requests are relayed to the intended recipient. Prominent examples of these systems include Psiphon [14], Tor [8], Lantern [21], Ultrasurf [1], and commercial or self-deployed Virtual Private Network providers, often using tools such as Shadowsocks [6], OpenVPN [5], and Wireguard [9].

Since the accessed content in such systems is encrypted, and the relay is usually not the originator of any censored content, these systems work to circumvent purely content-based censorship. Many network censors thus attempt to block access to at least the most prominent of these tools, for example by obtaining lists of the IP addresses of relays used by the tools, and blocking any connections to those destination addresses. Preventing such *enumeration* attacks while still allowing censored clients to access new proxies is the core of the proxy distribution problem.

2.2 Proxy distribution

The proxy distribution problem (also called the bridge distribution problem, in the context of Tor's unlisted bridge relays) refers to the problem of distributing or assigning proxies to clients in a circumvention tool. This involves balancing between several factors, but a central tradeoff is between distributing too many proxies, making it easy for a censor to enumerate and block them all; and distributing too few proxies, making it hard for clients to acquire working proxies. As an example, Tor assigns bridges to different distribution channels (such as email, Telegram, and domain-fronting systems) and applies rate limits within these channels based on accounts. CAPTCHAs and computational puzzles (proofs-of-work) [12] have also been proposed as mechanisms to rate-limit enumeration, with limited effectiveness. Privately-operated proxies and VPNs can be distributed through limited contacts or rely on monetary charges to raise barriers to enumeration.

Proximax. McCoy, Morales and Levchenko introduced Proximax [16], a system that attempts to maximize the availability of proxies through selective distribution. In Proximax, trusted clients issue invitations to the circumvention tool, and the distributor arranges clients into a tree structure according to whom they were invited by; new proxies are distributed through the tree according to trust levels. Branches of the tree accumulate trust while the assigned proxies are available, while blocking of proxies distributed to a branch reduces their trust.

Similarly, systems such as rBridge [26], Salmon [10], and Lox [24] also allow users to invite others to the tool after accumulating credit (when assigned proxies have high uptime) or suspicion (when assigned proxies become blocked). The precise details of assigning credit or suspicion based on social connections, and the level of detail of the social graph revealed to the distributor vary between these systems. However, all of these systems heuristically aim to maximize the span of time that a proxy is available against a censor that can participate in the scheme by controlling some clients.

Ephemeral Proxies. A more recent approach to preventing enumeration is the use of many ephemeral proxies, which do not remain part of the network on a consistent enough basis to bother to block. Examples of this approach include Snowflake [2] and Webtunnel, browser-based proxies that connects to the Tor network; uProxy [19]; Lantern's "Browsers Unbounded" project [22]; and academic systems such as SpotProxy [15] and MassBrowser [18].

2.3 ENEM19

Nasr *et al.* [17] introduced a game-theoretic framework (ENEM19) for analyzing proxy distribution schemes. In this framework, a censor controls some fraction of participating clients, can coordinate the actions of those clients, and can choose when or whether to block access to the proxies it discovers. The censor is assumed to have global knowledge and coordination of the state of its clients and the blocked proxies, but cannot discover proxies through other means. The system then evolves in time steps, with new blocking actions, new requests for proxies from clients blocked in the previous step, and new proxy distribution happening at each step. At each time step, every proxy assigns a utility to each client, and every client assigns a utility to each proxy; the censor computes a utility for its actions based on the amount of blocked traffic and number of proxies obtained in the time step. Then a distributor can compute an optimal distribution strategy for that time step, and the censor can compute an optimal blocking strategy for the time step.

The ENEM19 paper then evaluates several possible settings in which the framework could be applied. These include different levels of client and proxy arrival; possible utility functions for clients and proxies; and possible utility functions for strategies. Extensive simulations showed that even with a strong adversary, the optimal distribution scheme can provide better total availability and other metrics than the heuristic schemes of rBridge and Proximax. While the original framework investigates the effect of different arrival rates, and concludes that higher proxy arrival rates are key to performance, it does not explicitly model any existing ephemeral scheme, which include high *churn* among proxies. Additionally, the framework focuses primarily on insider enumeration attacks, and does not capture newer information about censor capabilities.

2.4 Censor Threat Models

In parallel to the work of Nasr *et al.*, other researchers and leaks have developed a more detailed picture of the architecture of national censorship systems, and how these impact the capabilities and strategies of censors. Two highly-studied examples of network censorship infrastructures include the so-called “Great Firewall of China” (GFW) [20] and the Russian “technical measures to combat threats” system (TSPU) [27], while very recent leaks have revealed information about the technical capabilities of the Tiangou Secure Gateway (TSG) deployed by Geedge Networks [13]. Whereas many clients of the GFW are obscured by (carrier-grade) Network Address Translation, TSPU and some deployments of TSG are positioned in-path with the ability to apply client state in their actions. This enables additional capabilities such as temporarily dropping all traffic between a pair of hosts or a specific three-tuple of source host, destination host, and port [27], or rate-limiting flows from a specific three-tuple [27].

In addition to enumeration via distribution channels, censors also increasingly have used *fingerprinting* to detect and block proxies. For example, anomalous TLS fingerprints based on certificate chains, supported ciphersuites, and other protoocl options have been used to identify and block circumvention proxies [2]. This method does carry the possibility of false positives, causing censors to misidentify and block non-circumvention servers as proxies, leading to collateral damage. In this case, the utility of a censor’s blocking action might also change.

In combination, these details motivate us to study the ENEM19 framework in a situation where there may be multiple, non-colluding censors. While each censor could have different utility functions for its actions, the distributor may not be able to determine the capability of the censor for a particular user. Thus the "optimal" assignment when considering a global censor may have overall lower performance than other distributor strategies when faced with censors of this nature.

3 Methodology

3.1 Simulation

To investigate our research questions, we modified the ENEM19 framework simulator developed by Kon *et al.* [15]. As in the original framework, each simulation proceeds in time steps. At each time step, some new proxies randomly join the system according to a poisson process with rate λ, and new clients randomly join the system with rate μ. Next the *censor* follows some strategy to block a set of proxies. Then the *distributor* computes an assignment for proxies to each user; some users may not be assigned to any proxies. After recording the resulting state of the system, any updates to the *environment* are computed (if necessary for the scenario being simulated) and the simulation proceeds to the next step.

We run each simulation for 360 time steps, and for each simulation setting we run 5 simulations to obtain statistically repeatable results. The code for our modified simulator is available at [11] and we hope it can serve as a basis for further research by other researchers and developers in proxy distribution as well.

Metrics. For each time step of each simulation, we record whether each user can reach some proxy, whether each proxy is blocked, and whether each unblocked proxy has assigned users. From these data we compute the following performance metrics:
Client Connectivity: The fraction at each time step, and on average across the simulation, of clients that can connect to some unblocked proxy
Proxy Availability: The fraction of proxies at each time step and on average across the simulation, of proxies that are unblocked.
Average Wait Time: The average number of time steps a client must wait to be assigned an unblocked proxy once all of its proxies are blocked.
Proxy Lifetime: For each proxy, in how many time steps was did it serve clients before being blocked, (and average across proxies for the simulation.)

In some simulations we include further measurements, as described in the appropriate sections below.

Censors. In total, we implement four censor strategies:
- *"aggressive":* This censor, as described by Nasr *et al.*, blocks each proxy learned by the censor agents immediately.
- *"optimal":* This censor determines at each step which of its known proxies to block based on "blocking score" (total number of clients blocked) balanced against a utility loss due to the change in utility assigned to censor agents by the distributor. We follow the utility function described in [17] (included in Appendix A for reference).
- *"zig-zag":* This censor, which we describe in more detail in Section 3.3.1, does not use an explicit utility function as defined in the Nasr, *et al* framework.
- *"profiling":* Similar to the zig-zag censor, this censor, which we describe in more detail in Section 3.3.2, has no explicit utility function.

Distributors. We also implement four distributor strategies. The "kind", "strict", and "anti-zig-zag" censors follow the model of Nasr, *et al.*. They assign a utility to each client for each proxy based on a weighted sum of proxy attributes, assign a utility to each proxy for each client based on a weighted sum of client attributes, and then compute a stable matching using the Gale-Shapley deferred acceptance algorithm. For these three distributors, we implement the utility functions given in [17] (and included in appendix A). The final distributor strategy, "snowflake", is described in more detail in Section 3.2; it models the real-world Snowflake system [2] deployed by the Tor Project and has no explicit utility function.

Environments. Similar to the "Ecosystem" parameters of Nasr *et al.*, the Environment determines the network environment in which the simulation is carried out; we implemented four such environments. The basic, "alive" environment, described in Nasr *et al.* includes a "birth period" of 60 steps with low fraction of censor agents and relatively high arrival rates of proxies, followed by a "stable" period with similar arrival rates for proxies and users, but no clients or proxies leaving the system. The snowflake simulations modify this environment as described in Section 3.2. The "collateral" environment models traffic analysis and non-circumventing clients and servers, as described in Section 3.3. Finally, the "multi-censor" environment, as described in Section 3.4, models a situation with multiple independent censors.

3.2 Modeling ephemeral distributors

Snowflake. Snowflike is a proxy distribution system deployed by the Tor Project [2]. Snowflake radically modifies the dynamics of proxy distribution by allowing arbitrary Internet users to become short-term proxies by running a proxy in their browser with a plugin or on Android with the Orbot Android app. Because proxies run on residential and mobile networks, connecting clients to proxies may require NAT traversal, with some clients and proxies running in "restricted" conditions that prevent them from connecting to another restricted host. Each Snowflake proxy supports only a single user, and they are matched through a central broker on a first-come-first-served basis, subject to NAT traversal compatibility. Since proxies are ephemeral, enumeration and blocking by participation are expected to be less effective, so Snowflake makes no attempt to limit such enumeration.

Required simulation modifications. In contrast to the basic setting of Nasr *et al.*, Snowflake expects and relies on churn among proxies and clients, and expects proxies to arrive at a rate similar to clients. To account for this we modify the simulator to include a much higher arrival rate for proxies, arriving at the same rate as clients in each time step. Furthermore, we add a random process in which some fraction c of both clients and proxies leave the network at each time step; Bocovych *et al.* [2] found that roughly half of proxies leave the network every 24 hours. When a client's proxy leaves the network, we consider the client (but not the proxy) to be blocked and start a new wait period tracking the client; when a client leaves the network, if it was waiting its wait time concludes, and otherwise its proxy becomes available for assignment again. When proxies leave the network, they are considered to have permanently become inactive, but in contrast, when clients join the system, we choose a previously active client with some probability μ_{ret}.

Furthermore, Snowflake has multiple client and proxy types: both clients and proxies may be restricted or unrestricted with respect to NAT traversal, and a small fraction of Snowflake proxies are run on "stable" servers in unrestricted networks. Thus we modify the simulator to randomly assign new clients to the restricted or unrestricted network condition with some probability ρ_{NAT} and (2) randomly assign new proxies as stable with some probability ρ_{STAB}, and any remaining new proxies to restricted or unrestricted condition with the same ρ_{NAT} proportion as clients.

The Snowflake distributor in our simulation takes the set of all waiting clients and available proxies in each time step and makes assignments as follows. First, it assigns as many restricted clients to unrestricted proxies as possible. Then it assigns as many restricted proxies to unrestricted clients as possible. Finally, if any unrestricted proxies remain, they are assigned to unrestricted clients. The "Optimal" censor in the ENEM19 framework uses a distributor's estimate of a particular client to a proxy in determining what proxies to block; to apply this censor's utility function in Snowflake, the utility of a client to a proxy is 0 if both are on restricted NATs, 0.5 if both are unrestricted, and 1.0 if they have opposite types.

Additional metrics and parameters. In addition to the metrics collected for all simulations, we also track metrics by client and proxy type in Snowflake simulations.

3.3 Modeling traffic analysis

To explore censor strategies that make use of analyzing the traffic between clients and proxies on the censor's network, we allow the censor at each time step to detect which clients in its network are communicating with a given server, and which servers a given client in its network is communicating with. To model possible errors in the detection, we include a "true positive rate" ρ_{TP} that models the probability the censor will detect the communication between a client and a proxy in each time step. In a network environment without client churn, we also include a fraction p_{OL} that models the probability that a client will be online and connected to its proxy or proxies in a given time step.

Additionally, because such traffic analysis may inadvertently identify traffic not associated with circumvention tool use, we introduce a set of N_{srv} innocent servers and N_{cli} innocent clients that never use circumvention tools. At the start of each simulation, each proxy client and innocent client are randomly assigned a set of innocent servers according to a power-law probability distribution. We then include a "false positive rate" ρ_{FP} that models the probability that one of these innocent connections will be incorrectly identified as circumvention traffic by the censor at any given step.

For both traffic analysis censors described in this subsection, we run simulations with multiple true positive rates (0.9 and 0.99) as well as false poitive rates (0.01, 0.005, and 0.001). In addition to the basic metrics described above we measure at each step the fraction of innocent servers blocked as well as the fraction of all connections to innocent servers that are blocked.

3.3.1 *Zig-zag censor.*

A "zig-zag" censorship attack, first described by Dingledine [7], works by identifying an initial set of P proxies, for example, through the distributor, and an initial set K of proxy clients, possibly empty. Then in each step, the set of clients is augmented by the detection of any clients connected to proxies in P, and the set of proxies is expanded by the detection of any proxies serving clients in K.

Recent document leaks show that the Tiangou Secure Gateway censorship apparatus suports a similar functionality, so it is an important and challenging threat to consider. We implement a zig-zag censor in our simulator using the traffic analysis functionality described above. At each step, the proxies currently assigned to the censor's agents are added to a list of proxies discovered in the previous ζ_{watch} steps. Then this list is augmented by detecting any proxies connected to clients discovered in the previous ζ_{watch} steps, and the list of clients is augmented by detecting and clients connected to those proxies. Finally, the censor blocks the set of proxies first discovered ζ_{watch} steps ago. We also implement the modified distributor objective function proposed by Nasr *et al.*, which adds a penalty to any potential (client,proxy) match proportional to the cardinality of the symmetric difference between the set of proxies at distance 2 from the proxy and the set of proxies connected to the client. In contrast to the Nasr *et al.* simulation, our traffic analysis functionality allows us to consider potential collateral damage caused by the censor, and the impact of imperfect detection of circumvention flows.

3.3.2 *Host Profiling censor.*

Wails *et al.* [25] describe a host-based traffic analysis approach to proxy enumeration. In this attack, a passive censor (who does not interact with the distributor at all) attempts to identify any circumvention tool flows leaving the censored region or network. Rather than blocking such flows, which may be difficult to accomplish in real time, and subject to significant collateral damage, the censor tracks which hosts participate in such flows. Hosts that exceed a threshold number of detected connections over a given observation window are classified as proxies, and can be blocked. This attack is more challenging to defend by proxy distribution methods, since the censor does not directly interact with the distributor and schemes that use the history of users to assign them to proxies will not properly capture the loss incurred by a proxy assignment.

We model this in our simulator by enumerating the clients in the censored region at each time step and counting the number of detected flows to each server (or proxy). We introduce two new parameters to model the operation of this censor. First, the profiling window $w_{profile}$ is the number of previous steps to take into account when considering which proxies or servers to block. A larger window can decrease the likelihood of collateral blocking, but also leaves proxies unblocked for longer periods. The second parameter is the profiling threshold $\tau_{profile}$, the required number of detected connections over the course of the window to block a host.

3.4 Modeling multiple censors

Multi-censor rationale. Most previous work on proxy distribution models the problem as a game between a single censor and the distributor. The censor can participate in the protocol to learn proxies, and can block proxies to reduce circumvention. Blocked proxies report this status to the distributor, who can then stop distributing the proxy address to clients. In practice, however, a distributor serves clients from a wide variety of censored regions. These censors will have different capabilities in terms of infrastructure, and different objectives; and may therefore not block proxies at the same time or rate. Thus a distributor may not know that a proxy is no longer reachable in some regions, and therefore make seemingly optimal assignments that leave clients without reachable proxies.

To model this situation, we modify the simulator to instantiate multiple censors, each with its own range of client addresses. We introduce a distribution D_{net} that describes the relative weights of these client ranges, and when adding new clients, assign them to censored regions according to this distribution. At each step, each censor makes a determination about which proxies to block in its region based on proxies distributed to its agents and traffic it can observe between clients in its region and external hosts. When a single censor blocks a proxy or server, connections between that region and the proxy are blocked and any circumvention clients whose connections are blocked will request new proxy assignments from the distributor. However, the distributor is not notified that a proxy has been blocked until it is blocked by all censors.

For each set of multi-censor experiment, we monitor both the overall proxy availability and client connectivity, as well as the proxy up-times, client wait-times, and client connectivity for each censored region independently.

	null		optimal		aggressive	
	over	equal	over	equal	over	equal
1	99.96%	97.67%	97.02%	91.04%	95.02%	95.01%
2	0.03%	1.95%	2.53%	6.80%	4.11%	4.11%
3	0.0001%	0.31%	0.37%	1.61%	0.72%	0.72%
4	0	0.05%	0.06%	0.40%	0.13%	0.13%
≥5	0	0.01%	0.01%	0.16%	0.03%	0.03%

Table 1: Fraction of non-zero Snowflake client wait times in each experimental condition

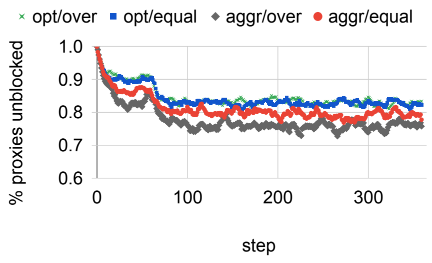


Figure 1: Snowflake proxy availability over time, in over-supplied and equal-supply scenarios. The aggressive censor blocks a higher fraction of proxies than the optimal censor

4 RQ1: Ephemeral Proxies

To understand how the "optimal" censor performs against ephemeral proxy distributors, we run multiple simulations of Snowflake under two different sets of network conditions:

In the "**over**provisioned" setting, proxies arrive at an average rate of 250 per time step, while clients arrive at an average rate of 200 per time step; this is greater arrival rate is consistent with the reporting of Bocovych *et al* [2], who reported, for example, a minimum of 80K unique proxy IP addresses per day in 2024 compared with an average of approximately 40K users per day.

In the "**equal**-arrival" setting, proxies and users both arrive at an average rate of 200 per time step. Because Snowflake always matches users to proxies on a 1:1 basis, any lower rate would lead to a permanent starvation of users.

We pair these settings with three censors: the *null* censor blocks no proxies, and provides baseline context for the rate at which clients arriving in a given round may not be able to pair with a reachable proxy; the *optimal* censor described in section 2, and the *aggressive* censor. In all simulations, we assume a 50% fraction of clients in the restrictive NAT condition, and 50% in the unrestricted condition, and 10% stable proxies. We model a 60-step "birth period" in which only 5% of new clients are censor agents, where as in the remaining step 10% are censor agents. Finally, we include a churn rate of 0.25 for users and proxies, and a "return" rate of 20% for users.

Figures 1 and 2 show the results of these simulations. In general, we can see that the "aggressive" censor outperforms the "optimal" censor in terms of both proxies and users blocked, but that the more important factor is the arrival rate of proxies relative to users: the "null" censor in the equal-arrival setting outperforms either censor in the over-provisioned setting. To provide further context, Table 1 shows that even among the fraction of users not receiving

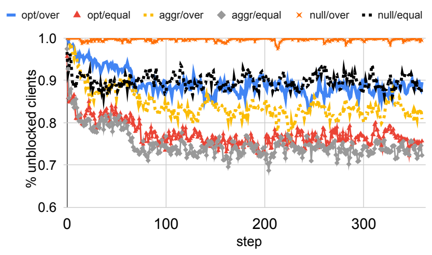


Figure 2: Snowflake client connectivity over time, in over-supplied and equal-supply scenarios. The "null" censor shows the effect of Snowflake's 1:1 proxy assignment.

w	τ	cls	U_{70}	U_{90}	S_{70}	S_{100}	S_{final}
3	3	best	2.41%	1.05%	99.54%	99.08%	96.82%
3	3	mid	2.45%	2.13%	97.86%	95.90%	85.82%
3	3	worst	1.71%	2.95%	96.01%	92.47%	73.09%
3	9	worst	2.43%	2.10%	98.74%	98.08%	94.54%
5	5	worst	0.62%	1.07%	96.88%	94.82%	82.56%
5	15	worst	96.61%	2.09%	99.04%	98.56%	95.98%

Table 2: Results of profiling attack against "kind" optimal distributor for various parameters. The "best," "mid," and "worst" classifiers are as described at the beginning of this section. U_n denotes the fraction of unblocked users at step n and S_n denotes the fraction of *unblocked* servers at step n.

an assignment on arrival, in over 90% of cases those users are assigned a proxy in the next step. Overall, we can see that **Snowflake effectively deters enumeration attacks**, with neither the aggressive nor the optimal censor preventing clients from connecting to proxies. We expect to see similar results for other distributors that change the parameters of the proxy distribution game by encouraging high rates of ephemeral proxy participation.

5 RQ2: Traffic Analysis

As discussed in section 3, to investigate how traditional proxy distribution algorithms perform against censors that may use traffic analysis, we model the ability of censors to monitor individual connections and employ rules to classify these connections as possibly being circumvention technology. We implement the "zig-zag" and "host-profiling" attacks and measure how they interact with distributor strategies. Because the results of both attacks can depend to some extent on the quality of the classification rules employed, in both sets of simulations we model three levels of classifier quality: the **best** classifier has true positive rate $\rho_{TP} = 0.99$ and false positive rate $\rho_{FP} = 0.001$; the **mid** classifier has $\rho_{TP} = 0.95$, $\rho_{FP} = 0.005$; and the **worst** classifier has $\rho_{TP} = 0.9$, $\rho_{FP} = 0.01$.

5.1 Zig-Zag attack

We measure how the ζ_{watch} parameter and the distributor functionality influence both the effectiveness of the circumvention tool over time, and the fraction of innocent servers, clients, and client-server connections that are collaterally blocked by this attack for the three classifier levels described above. In these simulations, we adopted the "Dynamic" environment of Nasr *et al.*, in which there is a 60-day "birth period" where users arrive at a rate of 20 per step, proxies arrive at a rate of 5 per step. and only 5% of users are censor agents; this is followed by an arrival rate of 10 users/step, 0.75 proxies per step, 10% of censor agents, and a proxy capacity of 40 users.

We first conduct a set of simulations using the "kind" optimal distributor and varying ζ_{watch}. In general, we expect that higher values of ζ_{watch} will lead to faster blocking, but also greater collateral damage, while larger ρ_{FP} will primarily increase collateral damage. Figure 3 shows the results of these simulations. We can see from Figure 3a that $\zeta_{watch} = 2$ is narrow enough that around 50% of users manage to connect to proxies despite the attack. Values of $\zeta_{watch} \in \{4, 6\}$ result very rapidly in almost total enumeration of the set of proxies and user blockage. However, as Figure 3b shows, all three settings also quickly incur high collateral damage, eventually blocking 50% of the innocent servers and their connections.

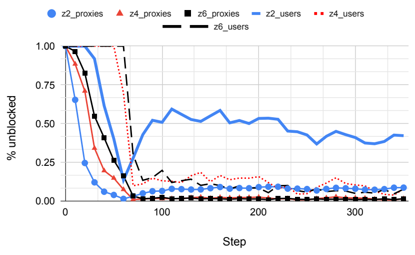
The next set of simulations explore the effectiveness of Nasr *et al.*'s proposed "Anti-Zig-Zag" optimal censor, and varying classifier quality; we hold ζ_{watch} constant at 4. We can see in Figures 4a and 4b that the quality of the classifier rule only slightly delays the time to maximal blocking, while significantly influencing the collateral damage incurred, with the lower-quality classifiers eventually blocking almost all innocent servers and connections.

Finally, given the possible resistance to Traffic analysis of Snowflake, we report on a set of simulations using the Snowflake distribution scheme. As Figure 4c shows, the ephemeral nature of clients and proxies in Snowflake prevents the zig-zag censor from effectively broadening their knowledge beyond the proxies learned by their agents. Thus we conclude that **previous proxy distribution algorithms defend poorly against zig-zag attacks**, but *ephemeral proxy schemes that avoid proxy re-use can effectively counter the zig-zag attack.*

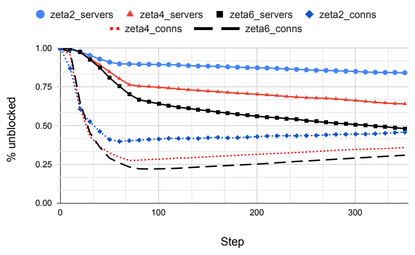
5.2 Profiling Censor

We measure how the $w_{profile}$ and $\tau_{profile}$ parameters and the distributor functionality influence both the effectiveness of the circumvention tool over time, and the fraction of innocent servers, clients, and client-server connections that are collaterally blocked by this attack. In general, we expect that lower values of w will lead to faster blocking, but also greater collateral damage; and higher τ will tend to reduce collateral damage. Similarly, higher values of ρ_{FP} should be expected to lead to worse collateral damage for a given w and τ.

To measure this, we conducted a set of simulations with the kind optimal distribution algorithm. In these experiments, we retain the "birth period" intended to allow the circumvention tool to grow quickly with minimal censor attention before starting censorship at higher activity at the end of the period. Since the profiling censor does not corrupt agents as in the other algorithms considered here, our simulation simply has the censor begin profiling at the end of the birth period, meaning that the censor will wait at least w steps before blocking any hosts. Thus we measure the fraction of unblocked clients and level of collateral blocking shortly after the end of the birth period as well as at the end of the simulation. Table 2 shows that the host profile attack is indeed a very strong network censorship strategy. Even for very low window sizes and relatively poor classifier rules, essentially all user traffic is blocked

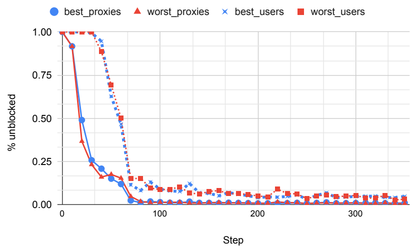


(a) User and Proxy blockage amounts for $\zeta_{watch} \in \{2,4,6\}$. Note the "birth period" of high proxy arrival and low censor agent arrival ends at step 60.

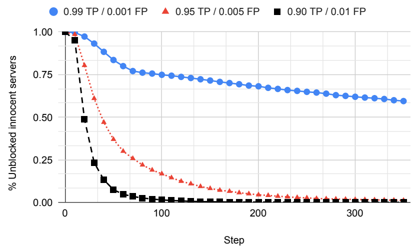


(b) Collateral Damage for $\zeta_{watch} \in \{2,4,6\}$. All experiments use the *best* classifier, $\rho_{FP} = 0.001$, $\rho_{TP} = 0.99$

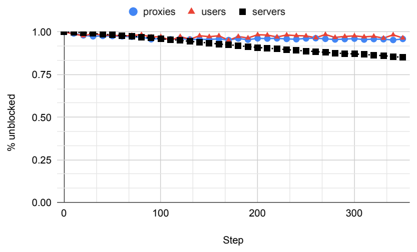
Figure 3: Zig-zag simulation results



(a) User and Proxy blockage amounts for highest-quality ("best") and lowest-quality ("worst") classifier, using the "Anti-Zig-Zag" distributor



(b) Collateral Damage for Zig-Zag censor using best, worst, and medium-quality classifiers, using "Anti-Zig-Zag" distributor



(c) Blocking effect and collateral damage of Zig-Zag attack against Snowflake. Reachability remains above 95% after 360 time steps, while collateral damage is also reduced.

Figure 4

t	1	2	3	4	5	6
Pr[$\leq t$],worst	84.25%	94.48%	98.01%	99.25%	99.72%	99.9%
Pr[$\leq t$],best	84.29%	94.50%	98.03%	99.29%	99.73%	99.9%

Table 3: Fraction of non-zero waiting times for Snowflake clients against Profiling Censor, $w = 3, \tau = 3$, *worst* and *best* classifiers

within 30 time steps. However, unlike in the zig-zag attack, host profiling has much lower collateral damage, never blocking more than 30% of the innocent servers in the simulations.

We also tested Snowflake against host profiling as applied in our model. Snowflake is more effective against the profiling censor, as Table 3 shows, with over 94.48% of users being assigned to a proxy within 2 steps, even with $w = 3, \tau = 3$ and regardless of the quality of the classifier. Furthermore, the final fraction of non-blocked servers in these simulations was 91.24% for the *worst* classifier and 99.04% for the *best* classifier. Thus we conclude that **even low-quality classifiers make effective host profiling attacks** and **more work is needed to develop the space of strategies that use and counter traffic analysis**.

6 RQ3: Multiple Censors

To study the effects of multiple, non-colluding censors with differing capabilities on proxy distribution algorithms, we ran multiple simulations in scenarios with two censored networks, and either a

censor	weight	proxy-use	90% wait	collateral
optimal	0.75	5	25	-
optimal	0.25	6	13	-
optimal	0.5	2	27	-
optimal	0.5	3	22	-
optimal	0.75	3	38	-
aggressive	0.25	3	8	-
optimal	0.5	2	25	-
aggressive	0.5	2	7	-
optimal	0.75	6	3	-
zigzag	0.25	4	10	44.40%
optimal	0.5	11	1	-
zigzag	0.5	4	4	90.72%

Table 4: Multi-censor simulations, "Anti-zig-zag" distributor. Proxy-use is the median span between first assignment and blocking, 90% wait is the 90th-percentile waiting time for a client. Each scenario aggregates results from 5 simulations.

25%/75% or 50%/50% split between the networks. In each simulation scenario, one of the networks was controlled by the optimal Nasr *et al.* censor, and the other was controlled by an optimal. aggressive, or zig-zag censor. We simulated the performance of the Anti-Zig-Zag distributor in each scenario.

Table 4 displays a summary of some of the results of these simulations. As expected, when both networks use the same censor strategy, outcomes are similar. However, imbalances in the networks (either in size or capabilities) can lead to suboptimal outcomes from the point of view of the censor, or from the point of view of the distributor. We conclude that **single-censor evaluation does not accurately predict more realistic deployment performance of distribution strategies.** Further work is needed to model the correct objectives in these scenarios.

7 Conclusions and Future Work

We have studied how changes in the capabilities of both Internet censors and circumvention tools impact the proxy distribution problem. By implementing a simulator that models ephemeral proxies and NAT restrictions, as well as traffic analysis capabilities we were able to study the behavior of the "optimal" strategies from previous work and show that in many cases these strategies (for censors and distributors) are not optimal. In many cases, the simple strategy of recruiting more proxies significantly improved outcomes compared to the more sophisticated mechanisms studied by Nasr *et al.*, while leaving room for future exploration of the problem.

An important problem and current bottleneck not captured by our simulator or the original framework is the design and selection of blocking-resistant control channels that allow distributors to communicate proxy assignments to users. The question of how to incorporate this aspect of circumvention into the framework (and the design of such channels) is an urgent one for future work.

We hope that our simulation framework can serve as a useful tool to explore additional questions raised by this work. For instance, while Snowflake performed well in resisting our simulated traffic analysis attacks, in practice the system has experienced service outages due to censors blocking flows rather than proxies. An important next question is what strategic capabilities for distributors would assist in evading such attacks. Another interesting problem to analyze further is how an optimal censor would combine traffic analysis with strategic use of compromised CT users; and how the interaction between different censors and these capabilities should influence the optimal distribution strategy for a longer time horizon. Finally, we did not investigate how social-graph based schemes such as Lox [24] or rBridge [26] are impacted by the censor actions incorporated in our framework; this is an interesting question for future work as well.

Acknowledgments

We thank Patrick Kon and the authors of [15] for sharing their implementation of the ENEM19 framework, and our anonymous shepherd for their helpful suggestions to improve the paper.

References

[1] Jacob Appelbaum. 2012. *Technical analysis of the Ultrasurf proxying software.* Technical Report. The Tor Project. https://media.torproject.org/misc/2012-04-16-ultrasurf-analysis.pdf
[2] Cecylia Bocovich, Arlo Breault, David Fifield, Serene, and Xiaokang Wang. 2024. Snowflake, a censorship circumvention system using temporary WebRTC proxies. In *USENIX Security Symposium.* USENIX.
[3] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. 2006. Ignoring the Great Firewall of China. In *Privacy Enhancing Technologies.* Springer, 20–35. https://www.cl.cam.ac.uk/~rnc1/ignoring.pdf
[4] Jedidiah R. Crandall, Daniel Zinn, Michael Byrd, Earl Barr, and Rich East. 2007. ConceptDoppler: A Weather Tracker for Internet Censorship. In *Computer and Communications Security.* ACM, 352–365. http://www.csd.uoc.gr/~hy558/papers/conceptdoppler.pdf
[5] OpenVPN developers. 2001. OpenVPN's Network Protocol. (2001). Published Online: https://build.openvpn.net/doxygen/network_protocol.html.
[6] Shadowsocks Developers. 2016. Shadowsocks. online: https://github.com/shadowsocks/shadowsocks/wiki. https://github.com/shadowsocks/shadowsocks/wiki
[7] Roger Dingledine. 2011. Research problems: Ten ways to discover Tor bridges. Online: https://blog.torproject.org/research-problems-ten-ways-discover-tor-bridges/.
[8] Roger Dingledine, Nick Mathewson, Paul F Syverson, et al. 2004. Tor: The second-generation onion router.. In *USENIX security symposium*, Vol. 4. 303–320.
[9] Jason A Donenfeld. 2017. WireGuard: Next Generation Kernel Network Tunnel.. In *NDSS.* 1–12.
[10] Frederick Douglas, Rorshach, Weiyang Pan, and Matthew Caesar. 2016. Salmon: Robust Proxy Distribution for Censorship Circumvention. *Privacy Enhancing Technologies* 2016, 4 (2016), 4–20.
[11] Hassan Fares, Omkar Fulsundar, and Nicholas Hopper. 2025. ProxySimulator. online: https://github.com/hoppernj/ProxySimulator.
[12] Nick Feamster, Magdalena Balazinska, Winston Wang, Hari Balakrishnan, and David Karger. 2003. Thwarting web censorship with untrusted messenger discovery. In *International Workshop on Privacy Enhancing Technologies.* Springer, 125–140.
[13] Lea Horne and Marla Rivera. 2025. *The Internet Coup: A Technical Analysis on How a Chinese Company is Exporting The Great Firewall to Autocratic Regimes.* Technical Report. InterSecLab. Available online: https://interseclab.org/research/the-internet-coup/.
[14] Sam Kean. 2007. Internet Research, Uncensored. *Chronicle of Higher Education* 53, 29 (2007).
[15] Patrick Tser Jern Kon, Sina Kamali, Jinyu Pei, Diogo Barradas, Ang Chen, Micah Sherr, and Moti Yung. 2024. SpotProxy: Rediscovering the Cloud for Censorship Circumvention. In *USENIX Security Symposium.* USENIX. https://www.cs-pk.com/sec24-spotproxy-final.pdf
[16] Damon McCoy, Jose Andre Morales, and Kirill Levchenko. 2011. Proximax: A Measurement Based System for Proxies Dissemination. In *Financial Cryptography and Data Security.* Springer. https://cseweb.ucsd.edu/~klevchen/mml-fc11.pdf
[17] Milad Nasr, Sadegh Farhang, Amir Houmansadr, and Jens Grossklags. 2019. Enemy At the Gateways: Censorship-Resilient Proxy Distribution Using Game Theory. In *Network and Distributed System Security.* The Internet Society. https://people.cs.umass.edu/~amir/papers/TorGame.pdf
[18] Milad Nasr, Hadi Zolfaghari, Amir Houmansadr, and Amirhossein Ghafari. 2020. MassBrowser: Unblocking the Censored Web for the Masses, by the Masses. In *Network and Distributed System Security.* The Internet Society. https://www.ndss-symposium.org/wp-content/uploads/2020/02/24340.pdf
[19] University of Washington. 2015. uProxy v1.2.5 – Design Doc. Online: https://docs.google.com/document/d/1t_30vX7RcrEGuWwcg0Jub-HiNI0Ko3kBOyqXgrQN3Kw/.
[20] John Palfrey, Jonathan Zittrain, Nart Villeneuve, Rafal Rohozinski, Derek Bambauer, and Ronald J Deibert. 2005. Internet Filtering in China in 2004-2005: A Country Study. *ONI Country Reports* (2005).
[21] Brave New Software Project. 2024. Lantern: better than a VPN. Online: https://lantern.io.
[22] Brave New Software Project. 2025. Browsers Unbounded. Online: https://unbounded.lantern.io.
[23] Sparks, Neo, Tank, Smith, and Dozer. 2012. The Collateral Damage of Internet Censorship by DNS Injection. *SIGCOMM Computer Communication Review* 42, 3 (2012), 21–27. http://conferences.sigcomm.org/sigcomm/2012/paper/ccr-paper266.pdf
[24] Lindsey Tulloch and Ian Goldberg. 2023. Lox: Protecting the Social Graph in Bridge Distribution. *Privacy Enhancing Technologies* 2023, 1 (2023). https://petsymposium.org/popets/2023/popets-2023-0029.pdf
[25] Ryan Wails, George Arnold Sullivan, Micah Sherr, and Rob Jansen. 2024. On Precisely Detecting Censorship Circumvention in Real-World Networks. In *Network and Distributed System Security.* The Internet Society. https://www.robgjansen.com/publications/precisedetect-ndss2024.pdf
[26] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas J. Hopper. 2013. rBridge: User Reputation based Tor Bridge Distribution with Privacy Preservation. In *Network and Distributed System Security.* The Internet Society.
[27] Diwen Xue, Benjamin Mixon-Baca, ValdikSS, Anna Ablove, Beau Kujath, Jedidiah R. Crandall, and Roya Ensafi. 2022. TSPU: Russia's Decentralized Censorship System. In *Internet Measurement Conference.* ACM.

A ENEM19 Utility Functions

We briefly recap the utility functions defined in [17]. The "Kind" and "Strict" distributors use the same utility functions, with different weights. For these distributors, the utility of a proxy to a user is given by $\beta_1 B + \beta_2 c + \beta_3 \tau - \beta_4 d$, where B is the total number of users that know the proxy, c is the current number of users of the proxy, τ is the lifetime of the proxy, and d is the user-proxy network latency. The utility of a user to the proxy is given by $\alpha_1 T - \alpha_2 R - \alpha_3 \gamma - \alpha_4 \delta - \alpha_5 d$, where T is the user's total proxy utilization (capped at some maximum value), R is the number of times the user has requested proxies, γ is the user's blocked proxy usage, δ is the number of blocked proxies assigned to the user, and d is the user-proxy network latency; the strict distributor assigns higher values to α_3 and α_4 while reducing the weight α_1. The "anti-zig-zag" distributor adds a sixth term to the user utility , $-\alpha_6 H$, where H is the number of proxies used by all other users of the proxy that were not used by the user. Finally, the "optimal" censor utility function defined in [17] is given by $\omega \sum_a U(a) + r_{\text{blocked}}$, where $U(a)$ is the utility assigned by the distributor to censor agent a, and r_{blocked} is the total number of users blocked, both given a particular set of blocking decisions.