GAME of DECOYS: Optimal Decoy Routing Through Game Theory by Milad Nasr and Amir Houmansadr

A Summary by Caleb Shere

May 18, 2017

1 Introduction

1.1 Internet Censorship

The Internet has been called the information superhighway. Today a child and a smart phone, with a few clicks and swipes, can access a breadth of information that dwarfs the world's greatest library. With such potential at the fingertips of the individual, naturally some people, particularly those in charge of authoritarian governments, seek to restrict it. In countries that protect the freedom of speech, citizens enjoy free access to the Internet with only a few extreme exceptions, but more authoritarian governments will block sites that merely promote diversity of political opinion. In China for instance, both Facebook and Youtube are blocked, and in fact China is one of the most prominent Internet censors, with its censorship mechanism being coined The Great Firewall of China. Naturally the citizen seeks to find a way to bypass restrictions of his Internet access, and a censoring government seeks to stop him from doing so. This paper discusses how each side achieves its goal and the conflict that emerges between them.

Before we get into the mechanics of Internet censorship, we best start with a brief description of how the Internet works. Put simply, the Internet is a network of networks, and a network is a collection of computers communicating with each other. These computers reside somewhere and belong to some person or corporation or entity of a sort. A single homeowner might control just a few devices connected to the Internet, while a network provider or other corporation may control hundreds or thousands or more. An Autonomous System (AS) is in essence a collection of connected devices under control of a single administrative entity. A country's network may consist of hundreds of Autonomous Systems. Finally, every device connected to the Internet has an IP address. Like a home address, a device's IP address is what other devices use to send data to this device. When one computer wishes to visit a website on some server, it sends a packet of data containing the IP address of the server, which is then passed along a series of routers, often through a number of ASes, until it reaches its destination.

In theory, there may exist quite a number of ways for a censoring authority to restrict his citizens' Internet access, but the most common way - and what our threat model deals with - is the use of an IP firewall. The firewall contains a blacklist of

forbidden IP addresses and blocks any request destined for those addresses. The exact implementation of the firewall varies, but it must be configured into a border router or some device that all outgoing traffic of a network passes through. When a packet reaches the device with the firewall, the firewall inspects the destination IP address of the packet, and if it is one of the addresses on the blacklist, the firewall blocks the packet from proceeding any further on its route. A capable censor can deploy the firewall on his entire country's network so that all people living in the country have restricted Internet access.

1.2 Outline

First, the paper discusses an effective method of circumvention called **decoy routing**. With decoy routing, the user sends a request that is addressed to some destination permitted by the censoring government, but this destination is, in fact, a decoy. When this seemingly innocuous request reaches a certain router *outside* of the censored network (a router specifically chosen and configured by the user and his cohorts) called a decoy router, the router instead sends the request to the forbidden destination which was where the user actually intended to get to. In this way, the user attains access to the restricted site.

Next, the paper discusses the censor's response to decoy routing. Decoy routing relies on a packet reaching a specific router (the decoy router) on its path to its destination. To destroy its effectiveness, the censor can change the paths that packets originating from his network will take through the Internet so that they always avoid decoy routers. This process is called **routing around decoys** (RAD).

But as we will see, neither one of these methods truly beats the other. Each comes with a cost - namely money and, in the case of RAD, changing Internet routes may weaken the censor's connectivity to the Internet as a whole, even the parts that he has no wish to restrict. In addition, the effectiveness of each method depends largely on where and how it is deployed. The last part of the paper analyzes, using game theory, how well each method performs in the face of the other. By the end, we will have a deeper understanding of the strengths - and weaknesses - of Internet censorship.

2 Decoy Routing

2.1 Background

Before decoy routing, the classic approach to circumvention was the use of a proxy server. But the important drawback of a proxy server is that once the authorities get wind of it, they can simply block it. The proxy is an entire endpoint on its own, so it has its own IP address, and therefore the censor can easily add this IP address to the blacklist in its firewall. The proxy is now just another blocked site that the client cannot reach.

Before we get to decoy routing, let us think about how we might fix the proxy server situation. There is a minor penalty that the censor endures for blocking the proxy, namely that an otherwise perfectly good server out there in the Internet is now inaccessible to the entire country. If not for its use as a proxy server, the censor would have happily kept his country connected to it. Granted, blocking one

additional site, or even a handful of sites, is no major blow to the censor. Whatever they provided can be attained elsewhere. But what if half of the Internet turned into a proxy service? Or better yet, all of it?

It is infeasible for the censor to block the entire Internet, as such action would isolate his country from the whole rest of the world, which would be severely damaging to its economy. The censor would then be in a tough state. Unfortunately, however, it is also infeasible to turn every computer in the world into a proxy server, so this solution seems out of reach. And here's where decoy routing comes in.

Decoy routing achieves the same functionality of our desired solution by placing the circumventing mechanism in a router rather than in a computer or other network host. A router is like a small computer that has one purpose - to receive packets and forward them to where they need to go. Routers connect networks to each other, as well as the computers within a network. Placing the circumvention mechanism in a router is important for two reasons. One, a packet contains the IP address of its final destination but *not* the address of all the routers it will meet along the way. The censor's firewall therefore cannot block a router like it does an endpoint. And two, the topology of the Internet is vast and dense so a well chosen router may reside along the path to millions of endpoints [1]. So with our circumventing router, the user can send a request to any of those millions of destinations and the router will instead give him access to a restricted site. The router has turned millions of computers into proxy servers!

2.2 How It Works

The first step is to set up a decoy router. This requires the cooperation of an Autonomous System that lies outside the censor's network. As the censor can only block requests entering or leaving his own network, once the user's request gets past the censor's firewall and reaches the decoy router beyond, the censor will not be able to stop the router from rerouting the request to a forbidden site. It is best to set up a number of decoy routers (which may require cooperation with multiple ASes), and it is best to choose them so that they lie on a path to as many permitted destinations as possible. The reasons for this will soon be clear.

Next, the client and the administrator of the decoy router must establish between themselves a secret code that will allow the client to signal the decoy router to reroute his connection. This is typically accomplished by inserting a specific string into a subtle location in a packet. One example is in the client random number field of the TLS handshake. TLS, or Transport Layer Security, is a cryptographic protocol that allows two hosts to communicate without their correspondence being read by third parties. To initiate a TLS session, the client sends a Hello message containing a random number field, used for establishing cryptographic secrecy, to the server. In this field the client can hide his string that signals the decoy router. Another example for a secret code is hiding the string somewhere in the user's cookie in an HTTP connection.

Alternatively, the code isn't even a secret message in the packet. The client can also signal the decoy router via "port knocking," by which he sends a series of packets destined to specific ports on the destination computer. The sequence of port numbers is the secret code that signals the router. Similarly, the client could send a series of packets with specific payload lengths, forming a number sequence that,

here too, signals the router. Ultimately, the client and decoy router can establish any code they want. The main requirement is that the router can easily identify and the censor can't [1].

The reader at this point might be wondering why the censor cannot simply identify those packets that contain secret messages and block them from exiting the censored network. After all, in order for decoy routing to be usable, people must know how to signal the decoy router, and in that case the censor is likely to know it as well. In fact, when we talk about routing around decoys later, we assume that the censor does know how to signal decoy routers which is what allows him to execute his counter attack. So why can't he just block packets that request decoy routing?

First, it is likely to make the censor's network significantly less efficient. As it is, the censor's firewall must inspect every packet that leaves the network and ensure that it is not destined for a forbidden destination. This takes a little bit of time for every packet, but it's extra time that the censor is prepared to spend. If, however, the firewall had to also check for decoy routing signals in every packet, it may slow things down more than the censor can afford. More significantly, if the client is using consecutive payload lengths or port numbers to signal the decoy router, then the censor would continuously have to store this information from the previous X number of packets sent by every client in the network. This would likely be infeasible for the censor.

Second, while the censor can inspect the IP headers of packets to find the destination IP address, he cannot always inspect the payload (the actual data). The client can use protocols (such as HTTPS and TLS) that implement end-to-end encryption. With only the decoy router and the client having the ability to decrypt the messages between them, the censor will not be able to identify decoy routing signals in packets sent by other people.

Now for step 3, when the client wants to connect to a forbidden site, he must send a request to a *permitted* site such that the path to that site contains a decoy router. He accomplishes this by inserting his secret string into a number of packets and sending them to various destinations, thus probing different Internet paths hoping to find a decoy router. When one of his requests reaches a decoy router, the router sends back a "hello" message that covertly tells the user that his flow is being connected to a forbidden site. From this message, the user knows he has found a decoy router. If he doesn't receive any such message, he tries connecting to a different site until he gets one. This explains why decoy routers should connect to as many endpoints as possible - the more sites that have a decoy router on their path, the easier it will be for a user to find one [1].

Once the decoy router detects an encoded packet, it forwards it to a proxy server that then completes the TLS handshake with the client (this assumes that the sentinel string that signals the decoy router was hidden in the TLS handshake). The proxy server uses data from the packets that the client sends to forge its own packets so that they look like they're originating from the decoy site. The client, by the end of the process, will be engaged in a TLS session with a proxy server that *looks* like a TLS session with the permitted decoy site. The proxy server then allows the client to connect to any site that he wishes [1].

An ending dose of clarity may be necessary here. You may be wondering why this process is better than a simple use of a proxy server if, in the end, that is what it accomplishes. Recall that the weakness of a proxy server is that its IP address

can be blocked by the censor just like every other restricted site. In the case of decoy routing, however, the user never addresses his request to the proxy server, but rather to any of millions of permitted sites. The proxy remains entirely hidden and out of the censor's control. Only the decoy router connects to the proxy, and this connection is executed outside of the censor's network. If the censor finds out that his rule was circumvented, he may block the IP address used, but the citizen can simply use a different IP address whose path connects to the decoy router. With the decoy router connected to a large number of sites, it would be unreasonable for the censor to block them all.

3 Routing Around Decoys

With the citizen executing decoy routing, the censor must accomplish three things in order to regain the upper hand:

1. He must detect that decoy routing is occurring. On the surface, decoy routing is undetectable.

2. He must find out where the decoy routers are. For the citizen, it is enough just to find a route that contains a decoy router, but the censor must determine the specific router on the route that is executing the circumvention.

3. He must alter the routing policies of his routers so that packets originating from his network follow a route that avoids decoy routers. This is called routing around decoys, or RAD.

It should be noted here that although routing around decoys is integral to the game that follows and it is the counter attack that we assume the censor uses, it is not the only way a censor can stop decoy routing. The censor can instead launch a denial-of-service attack on the router. To do this, he sends the router so many requests that the router becomes too busy processing them to be able to handle requests from other users. Alternatively, the censor might fragment the packets leaving his network to such a small size that the first fragment doesn't contain the complete TCP header. This would significantly hinder the decoy router's ability to redirect a TCP flow to the decoy proxy, as it would have to reassemble the fragments and maintain an immense amount of state in order to do so [1]. For the purposes of the game, however, we consider only the steps enumerated above and explain them in more detail in the following sections.

3.1 Step 1 - Detection

First, to determine that decoy routing is occurring, the censor can analyze the properties of the paths that packets *say* they're taking versus the paths they're *actually* taking. For example, the censor sees one of its clients make a connection to IP address X. Over time, the censor has collected data and knows the maximum packet size of the path to X as well as the average time it takes for a packet to get to X and back. If the client's packets are bigger than the maximum packet size or if they're taking too long or short a time to get to X and back, then it is likely that the client is using decoy routing and in fact not connecting to X at all [1].

3.2 Step 2 - Discovery

Next, the censor must discover where the decoy router is located. This is accomplished by acting as a client and sending lots and lots of decoy-routing requests to various sites and looking for that Hello message in return. The decoy router sends this Hello message to notify the client that it is decoy routing - unfortunately in this case, it also notifies the censor. If the censor does not receive the Hello message, it marks all of the ASes on this path as clean. If he does receive the message, he consults his list of clean ASes and whichever ASes are on this path but not on the list are candidates for holding decoy routers. If there is only one candidate from this path, then the censor can be certain that that AS is a decoy AS. If there are multiple candidates, then the censor can find paths containing clean ASes that intersect with each candidate. The candidate AS is clean if the new path does not a return a decoy Hello message, and it is dirty if it does. The censor may have to probe a large number of paths but ultimately it is reasonable using this method to isolate all of the decoy ASes [3].

3.3 Step 3 - Route Around Decoys

To understand how the censor routes around the decoys, we must explain the setting a bit further. As I mentioned in the Introduction, the Internet is made up of Autonomous Systems (ASes). Each AS uses the Border Gateway Protocol (BGP) to route its packets to other ASes to get them where they need to go, and each AS has a clearly defined routing policy. The routing policy determines the best path for a packet to follow to its destination. The "best" path depends on factors like how long it will take, as well as the AS's relationship with other ASes on the path. ASes are classified into three categories: providers, customers, and peers. Peer ASes route each other's packets for free, but a provider AS charges a fee in order to route the packets of a customer AS. Naturally, an AS prefers to use routes that won't cost money, and it defines its routing policy accordingly.

To follow its routing policy, every router stores two tables. The *routing table* contains information about packet routes that the router has collected. It collects this information from data in headers of packets that it receives over time. The router then chooses from this table the best route for each destination (based on factors mentioned above) and stores it in the *forwarding table*. When the router has a packet to send, it inspects its forwarding table to determine where to send it. As the router continues to receive packets, it may need to update its routing table which may also lead to updating the forwarding table because the best route has changed based on this new information [3].

So too, the censor can manually change what is considered the best route by reconfiguring the router. The censor provides his list of decoy ASes to the router, and the router chooses the best path that does not contain a decoy AS. With the packets following only routes that avoid decoy routers, the decoy routers are useless and the censor once again has achieved restriction of his citizens' Internet access.

4 The Game: Decoy Routing vs. RAD

4.1 Game Theory

Game theory studies the interactions of rational individuals each trying to achieve their own goals. It essentially asks the question, "In a game, what is the best move for each player to make?" But said "game" doesn't have to be a board game in your living room. It can be a real life scenario where, for example, a group of restaurant chains are each trying to choose the optimal location for their new restaurant in order to gain maximum revenue. Or, in a classic example, two prisoners each have to choose whether to stay silent or rat out his friend. If they both stay silent, they both serve 1 year. If they both rat out, they both serve 2 years. And if one stays silent while the other talks, the one who remained silent gets 3 years while the one who talked goes free.

In order to determine the optimal move for a player, we imagine the players trading turns, each one adjusting his decision as the other adjusts his, until the game reaches an equilibrium point at which each player has settled on his move because he has concluded that changing it would not benefit him. For example, in the prisoner dilemma, player 1 might choose to stay silent. Player 2, knowing this, then chooses to rat him out in order to go free. If player 1 could respond, he would now choose to also rat out in order to get 2 years instead of 3 years. We now let player 2 respond, but he has no wish to change his move and neither does player 1. The equilibrium point is where both players rat out their friend.

Every game has four components:

1. Players. These are the individuals making the decisions. They are assumed to be rational and self-interested.

2. Actions. These are the available actions that the players can choose to make. In the restaurant case, a player's set of actions is the set of all possible restaurant locations. In the prisoner dilemma, each prisoner has only two available actions - stay silent or confess.

3. Consequences. These are the negative results of the actions. In the restaurant case, the most likely consequence is the monetary cost of building the restaurant.

4. Payoffs. These are the positive results of the actions. The payoff to be gained in the restaurant case is the amount of revenue the player will gain by opening his restaurant in a given location.

The conflict between decoy routing and RAD is also a game, and it has its own players, actions, consequences, and payoffs.

4.2 Players

Our game has two players. The first is *censor*, a governing entity that seeks to censor its citizens' Internet access. And the second is *decoy*, who is some entity that provides decoy routing service to the people of *censor*'s country. For the purposes of the game analysis, we denote A_{cens} as the set of all ASes in *censor*'s country and

A_{free} as the set of all ASes outside of *censor*'s country. *decoy* will pay money to ASes in A_{free} in order to deploy decoy routing in them [2].

4.3 Actions

Censor makes one of two choices for every pair (AS_i, AS_j) where $AS_i \in A_{cens}$ and $AS_j \in A_{free}$. He can either use the regular BGP route to route packets from AS_i to AS_j (we denote this choice as a^{BGP}) or he can use an alternative route (we denote this choice as a^{RBGP}). *censor*'s final action in the game (A_c) is the set of all these choices:

$$A_c = \{a(P_{i,j}) | \forall P_{i,j} \in \mathbf{P}\}$$

where $\mathbf{P}$ is the set of all paths from A_{cens} to A_{free}.

Decoy's action in the game is the set of ASes in which he deploys a decoy router. This will be a subset of A_{free} and is denoted:

$$A_d = \{AS_1, ..., AS_k\}$$

4.4 Consequences

The method at **censor**'s disposal is of course RAD, but RAD results in two main costs for *censor*:

1. **Reduction in connectivity (C_1).** Changing the BGP route in order to avoid decoy routers may result in needing to use slower routes to get to permitted destinations. Or worse, there may not even exist an alternative route for every permitted destination, in which case that destination becomes unreachable to *censor*'s country.

2. **Monetary cost (C_2).** By changing routes, *censor* may be forced to use more expensive routes. For example, he may have to switch from a route with a peer AS to a route with a provider AS.

In this paper, these costs are denoted as C_1 and C_2 respectively. In the original paper, the authors list six different costs that *censor* faces, but I have simplified them into these two categories. I denote: $C_{censor} = C_1 + C_2$.

In addition, **decoy** faces the cost of money that needs to be paid to each AS in order to deploy decoy routing in it. The total cost for *decoy* is the sum of all the money he pays the ASes and is denoted C_{decoy}. In addition, we assume *decoy* has set aside a budget F and is willing to spend his whole budget for the sake of the game. So *decoy* is confined by the constraint $C_{decoy} \leq F$.

4.5 Payoffs

The goals of the two players are opposites. *censor* seeks to strengthen the censorship as much as possible (up to the amount of censorship defined by his censoring policy), and *decoy* seeks to weaken it as much as possible. The payoff for both players then is measured by the amount of censorship that remains at the end of the game, only that *censor* wants more of it and *decoy* wants less.

We measure the amount of censorship with the **censorship metric** and denote it as S. A value of 1 indicates complete censorship, meaning that no decoy routing

will succeed and the country's citizens will not be able to access any restricted sites. A value of 0, on the other hand, indicates completely free access to the Internet. The metric is calculated as a ratio of the amount of decoy routing deployed to the total amount of Internet traffic from A_{cens} to A_{free}. To estimate the amount of decoy routing deployed, the original authors took every pair of ASes whose route connecting them contains a decoy router, multiplied their sizes together (where size is the number of IP addresses in the AS), and added up all of the products. To estimate the total Internet traffic, the authors did the same thing but for all pairs of ASes whether or not they are connected by a decoy router. So formally, we have:

$$S = 1 - \frac{\Sigma_{AS_i \in A_{cens}} \Sigma_{AS_j \in A_{free}} ||AS_i|| * ||AS_j|| * \delta(R_{i,j})}{\Sigma_{AS_i \in A_{cens}} \Sigma_{AS_j \in A_{free}} ||AS_i|| * ||AS_j||}$$

where the δ function is defined as

$$\delta(R_{i,j}) = \begin{cases} 1 & R_{i,j} \text{ has at least one decoy AS} \\ 0 & R_{i,j} \text{ has no decoy ASes or } R_{i,j} = \emptyset \end{cases}$$

4.6 Utility Functions

A **utility function** is an important concept in game theory. Clearly, it is not sufficient for a player to simply choose the action that results in the highest payoff, as such an action may also result in an intolerable consequence. Nor should the player simply choose the action that minimizes the consequences, as such an action may result in little to no payoff. A player's utility function measures his overall satisfaction and is expressed in terms of both the consequences and the payoffs.

In *censor*'s case, his overall satisfaction depends on his personality. Some censors may care more about maintaining censorship, while others may care more about avoiding monetary cost. So *censor*'s utility function is denoted:

$$U_{censor} = S * B_0 - (C_1 B_1 + C_2 B_2$$

where (B_0, B_1, B_2) is the censor's **profile**. B_0 quantifies how much *censor* cares about the censorship metric S, B_1 how much he cares about C_1 (loss of connectivity), and B_2 how much he cares about C_2 (monetary cost).

In *decoy*'s case, his overall satisfaction is simpler. He just wants as little censorship as possible, but with the caveat that he can't spend more than his budget. His utility function is:

$$U_{decoy} = 1 - S \text{ subject to } C_{decoy} \leq F$$

The goal of each player in the game is to maximize his utility function.

4.7 Strategies

In this section, I talk about how each player finds his desired action - that is, the algorithm by which the player finds his optimal action given the opponent's action. In the simulations that follow, the players run their respective algorithms many times until they reach the equilibrium (where running the algorithm again results in no significant change in choice of action). The algorithms I soon describe are used to find the *current* optimal action given the opponent's most recent action. Only when they reach the equilibrium do the players arrive at the *mutually* optimal scenario.

Algorithm 1 Finding decoy's best response (game one)

$A_d \leftarrow \{\}$
while $C_{decoy} < \mathrm{F}$ **do**
  Sort ASes : For each AS_i compute its benefit contribution: $B^{AS_i}_{decoy} = \sum_{(AS_s, AS_d) \in \mathbb{P}} ||AS_s|| \cdot ||AS_d|| \mathbb{1}\{AS_i \in R_{s,d}\}$ Sort ASes by their benefits.
  Pick ASes: Choose the AS with the highest benefit and add to A_d.
  Update: Remove all the routes that contain the selected AS.
$A'_d \leftarrow \{\}$
while $C_{decoy} < \mathrm{F}$ **do**
  Sort ASes : For each AS_i compute its benefit contribution: $B^{AS_i}_{decoy} = (\sum_{(AS_s, AS_d) \in \mathbb{P}} ||AS_s|| \cdot ||AS_d|| \mathbb{1}\{AS_i \in R_{s,d}\})/C(AS_i)$ Sort ASes by their benefits.
  Pick ASes: Choose the AS with the highest benefit and add to A'_d.
  Update: Remove all the routes that contain the selected AS.
if $U_{decoy}(A_d) < U_{decoy}(A'_d)$ **then**
  Return A'_d
Return A_d

Figure 1: *decoy*'s greedy algorithm. Figure taken from [2]

4.7.1 *Censor*

Given *decoy*'s decision, *censor* aims to find the optimal action that maximizes his utility function. This is rather simple actually. All of the route actions are independent of one another, so for each path $P_{i,j}$, *censor* chooses a^{RBGP} if that results in a higher utility than a^{BGP}. Otherwise, he chooses a^{BGP}. To calculate his utility for a given path, he uses *decoy*'s most recent action A_d and some arbitrary values for all other path actions (a^{BGP} or a^{RBGP}) because the route actions are independent. Thus, *censor* performs two computations for every possible path, that is $2||\mathbf{P}||$ computations to get his optimal A_c.

4.7.2 *Decoy*

Given the censor's action A_c, *decoy*'s problem amounts to a *budgeted maximum coverage* problem. The general form of that problem is: given a collection S of sets with associated costs defined over a domain of weighted elements, and a budget L, find a subset $S' \subseteq S$ such that the total cost of sets in S' does not exceed L, and the total weight of elements covered by S' is maximized. In our case, the sets are ASes, the costs are the fees charged by the ASes, and the weight of a given AS is how beneficial it is to *decoy* (i.e. how significantly it will weaken the censorship if a decoy router is deployed there). This problem is NP-hard, but the authors of the original paper provide a greedy algorithm that achieves an $O(1 - \frac{1}{e})$ approximation. See Fig. 1 for the greedy algorithm.

This is a direct adaptation of a known greedy approximation algorithm for the budgeted maximum coverage problem, and it can be summarized as follows:

- First greedily choose the set of ASes with the highest benefit contribution (that would decrease the censorship metric the most if decoy routing was deployed in them).
- Then start over but this time choose the set of ASes with the highest benefit-to-cost ratio.

	B_0	B_1	B_2
T_1: QoS-cautions, wealthy	L	H	L
T_2: QoS-cautious, poor	L	H	H
T_3: QoS-ignorant, poor	H	L	H
T_4: Irrational	H	L	L

Figure 2: The censor profiles to be used in the game. H = high, L = low, Qo = quality of service

- Keep the set that results in higher utility.

4.8 Simulating the Game

The authors of the original paper simulated the game many times for many different scenarios. They estimated the cost of each AS, based largely on the AS's size and its relationship with the censoring country (an AS that is closely tied economically to the censor would charge more for decoy deployment because it is more damaging to it to be blocked by the censor). In addition, they tested six different censor profiles, but I have simplified this to four profiles to fit with the simplified costs used in this summary. The four profiles used in the simulations are explained in Figure 2.

A "wealthy" censor (T_1) has a lower B_2 attribute, as he cares less about incurring monetary cost. A "QoS-cautious" censor (T_1 and T_2) has a high B_1 attribute, as he cares very much about maintaining good quality of service, which is badly affected by reduced connectivity to the Internet. Additionally, the QoS-cautious censors are assumed to care less about the censorship metric. The "irrational" censor (T_4), on the other hand, cares about nothing but maintaining a high censorship metric.

The authors also modeled their simulations on four specific countries - China, Syria, Saudi Arabia, and Venezuela. A country like China, who possesses a high number of ASes, is able to maintain good access to the Internet even when routing around decoys, while a country like Syria, which only has four ASes, is less able to afford a drop in connectivity.

As would be expected, and as the simulations showed, the key variables that affect the outcome of the game are the size of *decoy*'s budget and the nature of *censor*'s profile. A higher budget means better circumvention. Similarly, the censor's profile clearly impacts his routing decisions.

Figure 3 displays the effects that *decoy*'s budget had on the censorship metric and on his utility function. With enough money, *decoy* can get S all the way down to near 0. Figure 4 displays the impact of *decoy*'s budget on the number of decoy routers he deployed during the game. The simulations represented by these graphs used China as the censor with a T_1 profile.

The effects of different censor profiles can be seen in Figure 5. For this table, the original authors used China and a fixed decoy budget in their simulations. As I mentioned earlier, the original paper uses six different costs for the censor while I have simplified them to two. To calculate the quantities in my simplified table from

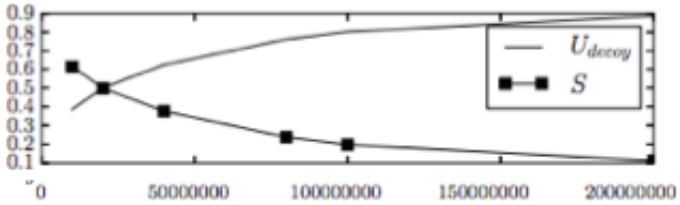

Figure 3: Impact of *decoy*'s budget on his utility and the censorship metric. The x-axis measures the size of the budget. Figure taken from [2]

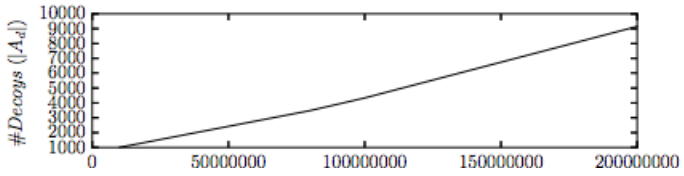

Figure 4: Impact of *decoy*'s budget on the number of decoy routers deployed. The x-axis measures the size of the budget.

the table in the original paper, I took the sum of the three costs that fall under "connectivity" and let that be C_1, and I did similarly for C_2.

Last, the country itself affects the game results. Different countries control more or fewer ASes, and the layout of these ASes and their routes can vary widely. Figure 6 shows the results of game simulations based on four different countries. The simulations used T_1 for each country and a fixed decoy budget.

4.9 Conclusion

The results of the game are generally good news for champions of freedom. Many had thought that RAD was a sure defense against decoy routing and that citizens would have to find another way to circumvent censorship. But as Figures 3 and 5 show, given enough money, the decoy player can achieve a censorship metric as low as 0.229 against a country like China, provided that the Chinese censor is rational and tries to avoid damage to himself. In Figure 5 we see that citizens of less well-connected countries like Saudi Arabia and Syria can achieve even better results. Granted, a more determined censor like profile T_3 can still keep the censorship metric high at .949, but this comes at the heavy price of cutting himself off from large portions

	C_1	C_2	Cen. Metric	# decoys
T_1: QoS-cautions, wealthy	0.000	0.087	0.277	2789
T_2: QoS-cautious, poor	0.000	0.009	0.229	2706
T_3: QoS-ignorant, poor	0.782	0.009	0.949	2710
T_4: Irrational	0.799	.093	1.000	2702

Figure 5: Impact of the censor profile on the costs incurred by the censor, on the censorship metric, and on the number of decoy routers deployed by *decoy*. These results are based on China as the censor and a fixed decoy budget.

	C_1	C_2	Cen. Metric	# decoys
China	0.000	0.087	0.277	2789
Venezuela	0.002	0.127	0.210	1450
Saudi Arabia	0.002	0.161	0.197	1853
Syria	0.003	0.011	0.101	544

Figure 6: Comparing results of various countries, using profile T_1 and a fixed decoy budget

of the Internet and incurring heavy monetary costs. A completely irrational censor (profile T_6) can still achieve perfect censorship by cutting himself off completely. This, however, is not specific to decoy routing. In the face of any circumvention technique, an irrational censor can disconnect from the Internet entirely and still maintain censorship. All in all, even with the censor routing around decoys, a reasonably wealthy and capable entity can generally achieve effective circumvention through decoy routing [2].

References

[1] Josh Karlin, Daniel Ellard, Alden W Jackson, Christine E Jones, Greg Lauer, David Mankins, and W Timothy Strayer. Decoy routing: Toward unblockable internet communication. In *FOCI*, 2011.

[2] Milad Nasr and Amir Houmansadr. Game of decoys: Optimal decoy routing through game theory. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1727–1738. ACM, 2016.

[3] Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing around decoys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 85–96. ACM, 2012.