Recruiting New Tor Relays with BRAIDS

Rob Jansen Nicholas Hopper Yongdae Kim

University of Minnesota
Minneapolis, MN 55455 USA
{jansen, hopper, kyd}@cs.umn.edu

ABSTRACT

Tor, a distributed Internet anonymizing system, relies on volunteers who run dedicated relays. Other than altruism, these volunteers have no incentive to run relays, causing a large disparity between the number of users and available relays. We introduce BRAIDS, a set of practical mechanisms that encourages users to run Tor relays, allowing them to earn credits redeemable for improved performance of both interactive and non-interactive Tor traffic. These performance incentives will allow Tor to support increasing resource demands with almost no loss in anonymity: BRAIDS is robust to well-known attacks. Using a simulation of 20,300 Tor nodes, we show that BRAIDS allows relays to achieve 75% lower latency than non-relays for interactive traffic, and 90% higher bandwidth utilization for non-interactive traffic.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Algorithms, Security

Keywords

Anonymous Communication, Peer-to-Peer Networks

1. INTRODUCTION

Tor [53] uses Onion Routing [24] to create a practical system for low-latency anonymity [14]. Tor *clients* periodically retrieve a list of *relays* from the Tor *directory service* and connect to Internet services by relaying requests through a *circuit* of multiple relays chosen from the downloaded list. The aggregate bandwidth costs of sending communication securely through multiple relays are significantly higher than direct communication: the amount of bandwidth expended by a client is also expended by each relay in its circuit. A significant characteristic of communication over Tor is that *most clients use Tor for interactive applications* like web browsing, but *most data is transferred for non-interactive applications* like file sharing [32]. Moreover, Tor relays forward traffic for multiple circuits simultaneously, further increasing bandwidth obligations. The combination results in overloaded relays and drastically increased latency for both interactive and non-interactive communication [32].

A lack of incentives to run relays combined with the associated costs has hindered relay enlistment, and in turn, Tor's scalability. Although relaying traffic can increase user anonymity by frustrating attempts to differentiate relay-sourced from relay-forwarded data, there are no measurable benefits to providing service for others. Consequently, clients greatly outnumber relays in Tor. In 2009, there were an estimated 100,000 simultaneously active Tor clients [31] but only about 1,500 Tor relays.[1] This uneven distribution of bandwidth responsibilities combined with the disproportionately high client-to-relay ratio results in poor system performance and a *tragedy of the commons* [26] scenario: as Tor grows, it will require additional relays to provide bandwidth and traffic forwarding services to remain usable.

Recruiting New Relays A significant problem faced by the current Tor system is *how to recruit new relays* to support expansion and ease the load suffered by current relays. There have been few approaches to solve the relay recruiting problem. One approach is to simply require every client to also be a relay, effectively reducing the client-to-relay ratio to 1:1 [42]. While we wish to promote relaying traffic, we do not wish to forcefully impose it: clients who are unable to run a relay due to censorship [48] would not be able to effectively use the system. Denying anonymity to clients in censored regions not only opposes the "anonymity for all" ideology, but also decreases anonymity for others since it reduces the diversity and size of the anonymity set – the set of potential circuit initiators. Tor's approach thus far has been to build a community and educate users about the benefits of anonymity, while simplifying relay setup and maintenance procedures. While this approach has been effective at expanding the network to its current size, relays are still in high demand and performance remains poor.

Introducing BRAIDS In this paper we present BRAIDS,[2] a set of practical mechanisms for the Tor anonymity network that increases incentives for relays while limiting the delays

[1]The corresponding client-to-relay ratio is 66:1.

[2]BRAIDS stands for Bandwidth Reciprocity And Incentivized Differentiated Services.

CCS'10, October 4–8, 2010, Chicago, Illinois, USA.

caused by non-interactive BitTorrent clients and keeping the system usable for everyone. Relays using BRAIDS enjoy lower latency and higher throughput than other users. In particular, BRAIDS allows relays to achieve 75% lower latency than non-relays for interactive web traffic – a 40% improvement over the current Tor network. Relays initiating non-interactive traffic receive a 90% increase in total bandwidth utilization from non-relay users.

To improve performance, BRAIDS incorporates differentiated services and a scheduler based on the proportional differentiation model introduced by Drovolis *et al.* [16, 17, 18, 19]. BRAIDS aggregates traffic into three hierarchical service classes *proportionally prioritized* as low-latency > high-throughput > normal, where the "cost" of high-throughput > low-latency (normal service is free). Each relay rate-limits the low-latency class to prevent high-throughput nodes from overwhelming low-latency traffic. Finally, traffic is paid and proportionally prioritized in *both* directions through the circuit, capturing Tor's asymmetric bandwidth requirements.

BRAIDS users optionally and anonymously "pay" relays with generic tickets that are both distributed freely in small amounts to all clients and relays, and collected by each relay while volunteering bandwidth to Tor. We use *relay-specific tickets* [29, 41] – random numbers combined with relay-identifiers – that are signed by an authority. Signed tickets are verified at the relay, defeating the double spending problem in which clients must make immediate deposits to catch cheaters that duplicate and spend a ticket multiple times. Information leakage is avoided since relays can verify tickets without assistance from an external entity. Tickets are valid during uniform intervals to prevent linking clients with tickets. Clients who cannot or choose not to pay receive slightly reduced performance.

Other incentive-based recruitment approaches exist in the literature: the gold star scheme [38] gives preferential treatment to fast relays whereas PAR [3] and XPay [10] use e-cash and an online bank to produce monetary incentives. A variety of attacks [20, 28, 33, 35] make it difficult to design a secure solution with minimal loss of anonymity. In particular, bandwidth accounting mechanisms that give better service to relays that volunteer more bandwidth [38] in some cases significantly decrease the anonymity set of relays receiving better service, and in others [3] unintentionally allow an adversary to link relays to the same circuit.

BRAIDS is secure, retaining all of Tor's anonymity for users browsing the web, whereas the previously proposed gold star scheme [38] achieves less than 65%. Our anonymous ticket approach mitigates the intersection attack that has plagued previous schemes. Further, BRAIDS bounds cheating in such a way that users must volunteer a significant amount of bandwidth before maliciously gaining an insignificant number of tickets.

Outline The remainder of the paper is outlined as follows. In Section 2, we briefly discuss BRAIDS system requirements while detailing the design in Section 3. Analysis of security and parameters is given in Section 4, while simulations and results are described in Section 5. Finally, Section 6 discusses related work and Section 7 concludes.

2. REQUIREMENTS

BRAIDS' main goal is to encourage Tor clients to run relays by providing incentives in the form of increased performance. This system should prioritize low-latency traffic over high-throughput traffic to reduce the negative impact that file sharing users have on overall system performance while remaining usable by everyone. The service received by web browsing clients should not reduce their anonymity.

BRAIDS shares the same threat model as Tor – a local adversary who cannot observe or interfere with traffic sent between honest nodes. While we do not defend against current attacks on Tor, our system should not reduce Tor's security by introducing any new vulnerabilities. We should not leak information about the circuit initiator or the identities of relays composing the circuit.

In addition to the aforementioned entities, we introduce a centralized, partially-trusted, offline bank to manage and certify bandwidth accounting tasks. The bank should only be trusted to follow protocol, but we assume it can otherwise attack the system using any information in its possession. BRAIDS should provide accounting mechanisms for both the outgoing path from client to server, and the reverse path from server to client (previous systems [3, 10] do not provide payment mechanisms for the reverse path of a two-way communication channel), since many existing applications (e.g. web browsing and streaming media) have significantly higher downstream than upstream client requirements. Bandwidth accounting should be anonymous to protect the client's identity, while payments must be unforgeable, non-reusable, and should not be linkable to the client [9, 54]. Additionally, we require double spending prevention in the form of immediate double spending detection. Clients attempting to double-spend should not receive service. Any attempts to cheat the system should be bounded so that the overall efforts required to cheat will outweigh the achievable benefits.

Finally, our system should be an incrementally deployable extension to Tor: users transitioning from legacy software should not be partitioned from the network.

3. SYSTEM DESIGN

BRAIDS motivates users to operate Tor relays by introducing generic tickets for service accounting. Using blind signatures, users remain anonymous while obtaining a limited amount of free tickets from the bank. Tickets are then embedded into Tor cells to request the desired class of service – either low-latency and low-throughput (e.g. general web browsing) or high-latency and high-throughput (e.g. downloading or sharing large files). Each relay verifies its tickets to prevent double spending.

3.1 Relay-specific Tickets

Our ticket design draws upon ideas from coin ripping [29] and fair exchange for mix-nets [41]. Since tickets are relay-specific, our construction requires that clients have *a priori* knowledge about their desired communication partners [43]. Tor already requires knowledge of relays when building circuits, so relay-specific tickets are an appropriate choice.

Ticket Structure A ticket T consists of a main part T_s, called the ticket *stub*, and a receipt part T_r, called the ticket *receipt*. The ticket stub contains the identity of the relay $\mathcal{R}$ (its public key) to which the ticket may be transferred. Letting $|$ denote concatenation, we define a ticket for $\mathcal{R}$ as:

$$T^{\mathcal{R}} = \{T_s^{\mathcal{R}} \mid T_r^{\mathcal{R}}\} = \{\mathcal{R} \mid H(T_r^{\mathcal{R}}) \mid d \mid \sigma \mid T_r^{\mathcal{R}}\}$$

where H is a cryptographically secure one-way hash function, d is a set of date-stamps, σ is the bank's partially

blind signature on $\{\mathcal{R} \mid H(T_r^{\mathcal{R}}) \mid d\}$, and $T_r^{\mathcal{R}}$ is a random bit-string used as a receipt.

Ticket Activation We use a blind signature scheme [8] to activate tickets and ensure no information about relay $\mathcal{R}$ chosen by client $\mathcal{C}$ is revealed. Specifically, our construction uses a partially blind signature [1] where the client blinds information about the chosen relay $\mathcal{R}$. The bank attaches uniform public date-stamps (described below) to the ticket, but cannot discover the blinded relay information. The bank's signature creates a strongly unforgeable ticket $T^{\mathcal{R}}$, i.e. modifying the signed contents invalidates the ticket.

Ticket Validity Intervals The bank attaches a set of date-stamps $d = \{d_u \mid d_v \mid d_w\}$ to the blinded relay information before signing. The time from ticket generation until the first date-stamp specifies the *spending interval* [-,d_u) in which relay-bound tickets may be spent. The time between the first and second date-stamp specifies the *relay-exchange interval* [d_u,d_v) in which a relay may exchange tickets at the bank for new relay-bound tickets. The time between the second and last date-stamp specifies the *client-exchange interval* [d_v,d_w) in which any client or relay may exchange tickets at the bank for new relay-bound tickets. Finally, tickets expire and are completely void after the final date-stamp. We suggest values for these parameters in Section 4.1.

The relay-before-client exchange priority prevents a client from maliciously exchanging a spent ticket before the relay can (causing the relay's ticket to appear double-spent upon attempted exchange) while still allowing the client to exchange unspent tickets. The final date-stamp prevents the bank's ticket database from growing infinitely large. The bank's global date-stamps are used for every ticket signed during a given time period to prevent the bank from marking tickets and linking clients with relays.

3.2 Ticket Transferability

Users may wish to transfer tickets to other users, or update tickets that have passed their spending interval but are not yet void. Unspent tickets may be transferred to relays for payment, but spent tickets or tickets past their spending interval must be first exchanged at the bank.

Users remain anonymous by exchanging tickets with the bank through Tor. However, in order to exchange during the relay-exchange interval, a relay is required to prove knowledge of its private key to the bank. Although this means relays are not anonymous in the exchange, we note that the bank can already enumerate the list of relays by downloading the public directory. The bank validates that the relay is bound to the exchanged tickets.

Relay Ticket Exchange When relay $\mathcal{C}$ receives ticket $T^{\mathcal{C}}$, it becomes a *voucher* for $\mathcal{C}$ redeemable for a new relay-specific ticket. Relay $\mathcal{C}$ and bank $\mathcal{B}$ use Protocol 1, `Relay-Ticket-Exchange`, to generate a new ticket spendable at relay $\mathcal{R}$ given that $\mathcal{C}$ presents a valid ticket voucher $T^{\mathcal{C}}$ and new ticket material. $\mathcal{C}$ performs setup on lines 1–2 by generating a random value and its hash. $\mathcal{C}$ sends $\mathcal{B}$ the voucher $T^{\mathcal{C}}$, and $\mathcal{B}$ is responsible for validating $T^{\mathcal{C}}$. $\mathcal{B}$ does this by verifying $T^{\mathcal{C}}$ is within the allowable date interval for relay-exchange, the identity $\mathcal{C}$ from $T^{\mathcal{C}}$ matches the real identity $\mathcal{C}$, $\mathcal{C} \mid H(T_r^{\mathcal{C}})$ never appeared before (i.e. $T^{\mathcal{C}}$ was not double spent), σ is a valid signature on $\mathcal{C} \mid H(T_r^{\mathcal{C}})$, and that a freshly computed $H(T_r^{\mathcal{C}})$ matches the hash from $T^{\mathcal{C}}$. If the voucher validates, $\mathcal{B}$ and $\mathcal{C}$ cooperate to produce a partially blind signature on the new ticket $T^{\mathcal{R}}$ payable to relay $\mathcal{R}$.

Protocol 1 `Relay-Ticket-Exchange`: $T^{\mathcal{C}}$ for $T^{\mathcal{R}}$ between relay $\mathcal{C}$ and bank $\mathcal{B}$ for service at relay $\mathcal{R}$. The arrows represent anonymous communication in Tor. The partially blind signature ($pbs\text{-}sign(\cdot)$) and verification ($pbs\text{-}verify(\cdot)$) are defined in [1].

Setup:

1: $\mathcal{C}$: generate random receipt $T_r^{\mathcal{R}}$
2: $\mathcal{C}$: construct partial stub $\overline{T_s^{\mathcal{R}}} = \{\mathcal{R} \mid H(T_r^{\mathcal{R}})\}$

Execution:

3: $\mathcal{C} \rightarrow \mathcal{B}$: redeemable voucher $T^{\mathcal{C}}$
4: $\mathcal{B}$: validate voucher $pbs\text{-}verify(T^{\mathcal{C}})$
5: $\mathcal{B}$: global ticket validity date-stamps $d = \{d_u \mid d_v \mid d_w\}$
6: $\mathcal{B} \leftrightarrow \mathcal{C}$: partially blind-signature $\sigma = pbs\text{-}sign(blind(\overline{T_s^{\mathcal{R}}}) \mid d)$
7: $\mathcal{C}$: construct full stub $T_s^{\mathcal{R}} = \{\overline{T_s^{\mathcal{R}}} \mid d \mid \sigma\}$
8: $\mathcal{C}$: construct ticket $T^R = \{T_s^{\mathcal{R}} \mid T_r^{\mathcal{R}}\}$

Protocol 2 `BRAIDS-Communication`: Message M from Client $\mathcal{C}$ to server $\mathcal{S}$ through relays $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$.

Setup:

1: $\mathcal{C}$ obtains tickets $T^{\mathcal{R}_i} = \{T_s^{\mathcal{R}_i} \mid T_r^{\mathcal{R}_i}\}$, for $i \in [1,3]$
2: $M_{\mathcal{C}\rightarrow\mathcal{R}_3} = E_{K_{\mathcal{C}\mathcal{R}_3}}\{T_r^{\mathcal{R}_2} \mid T_s^{\mathcal{R}_3} \mid T_r^{\mathcal{R}_3} \mid \mathcal{S} \mid M_{\mathcal{C}\rightarrow\mathcal{S}}\}$
3: $M_{\mathcal{C}\rightarrow\mathcal{R}_2} = E_{K_{\mathcal{C}\mathcal{R}_2}}\{T_r^{\mathcal{R}_1} \mid T_s^{\mathcal{R}_2} \mid \mathcal{R}_3 \mid M_{\mathcal{C}\rightarrow\mathcal{R}_3}\}$
4: $M_{\mathcal{C}\rightarrow\mathcal{R}_1} = E_{K_{\mathcal{C}\mathcal{R}_1}}\{T_s^{\mathcal{R}_1} \mid \mathcal{R}_2 \mid M_{\mathcal{C}\rightarrow\mathcal{R}_2}\}$

Execution:

5: $\mathcal{C} \rightarrow \mathcal{R}_1 : M_{\mathcal{C}\rightarrow\mathcal{R}_1}$
6: $\mathcal{R}_1$: verify $T_s^{\mathcal{R}_1}$
7: $\mathcal{R}_1 \rightarrow \mathcal{R}_2 : E_{K_{\mathcal{R}_1\mathcal{R}_2}}\{M_{\mathcal{C}\rightarrow\mathcal{R}_2}\}$
8: $\mathcal{R}_2$: verify $T_s^{\mathcal{R}_2}$
9: $\mathcal{R}_2 \rightarrow \mathcal{R}_1 : E_{K_{\mathcal{R}_1\mathcal{R}_2}}\{T_r^{\mathcal{R}_1}\}$
10: $\mathcal{R}_2 \rightarrow \mathcal{R}_3 : E_{K_{\mathcal{R}_2\mathcal{R}_3}}\{M_{\mathcal{C}\rightarrow\mathcal{R}_3}\}$
11: $\mathcal{R}_3$: verify $T_s^{\mathcal{R}_3}$
12: $\mathcal{R}_3 \rightarrow \mathcal{R}_2 : E_{K_{\mathcal{R}_2\mathcal{R}_3}}\{T_r^{\mathcal{R}_2}\}$
13: $\mathcal{R}_3 \rightarrow \mathcal{S} : M_{\mathcal{C}\rightarrow\mathcal{S}}$

Client Ticket Exchange Since a client might obtain tickets for a relay who is offline for the duration of the ticket's spending interval, a client may exchange a ticket for another bound to a new relay. The `Client-Ticket-Exchange` protocol (not shown) is essentially identical to Protocol 1, except that on line 4 the bank checks that the ticket is in the correct interval for client-exchange, but does not (and cannot) check for the identity of the client in the ticket.

Incorporating Tickets into the Tor Protocol Our ticket construction from Section 3.1 enables us to easily embed ticket stubs and receipts in Tor messages (i.e. cells). As shown in Protocol 2, `BRAIDS-Communication`, the client constructs the Tor message such that each relay on the path receives its own ticket stub and the receipt for the previous-hop in the path. There are two exceptions: the first-hop relay does not send a receipt to the client, and the last-hop relay receives a complete ticket (there is no next-hop relay).

We must include accounting mechanisms not only for the forward path from client to server, but also the reverse path from server to client due to asymmetric bandwidth requirements (e.g. streaming media). Since the reverse path cannot be paid by the server, clients pre-pay circuits (several cells can be transferred for each ticket) and relays notify clients when their paid balance expires. Clients embed tickets in outgoing cells using Protocol 2, which distributes tickets to each relay in the circuit. A relay lowers a circuit's priority when not paid and restores it after new tickets arrive. Since scheduling decisions are made locally and independently,

clients may choose to pay a subset of relays in the circuit without affecting scheduling decisions made by other relays.

Relays drop circuits upon detection of malicious activity, including forged tickets, and will only forward messages if a receipt is returned by the next-hop relay. Each relay is encouraged to participate faithfully to continue accumulating tickets since malicious activity stops the flow of tickets for all relays in a dropped circuit.

Double Spending We have shown that relay-specific payments eliminates the trade-off between double spending prevention [39] and information leakage, suffered by PAR [3], since tickets can be verified by the relay without a third party. Anonymous payments are appropriate in Tor since they protect the identity and privacy of the user.

3.3 Randomized Ticket Distribution

A major problem with the gold star incentive scheme [38] is that gold star relays can be distinguished from normal relays, since their gold star status appears in the public directory. This reduces their anonymity set – an adversary can be confident that if a client is receiving gold star service, that client also runs a relay.

To mitigate this problem, we assign each client *ticket distribution agents* – guard nodes that assist in distributing free tickets to all clients. We note that each Tor client already uses a small set of guard nodes from which a circuit entry node is selected to limit client identity (IP address) exposure to malicious entry nodes. Each agent distributes tickets from the bank to clients in proportion to the bandwidth it provides as a relay: the client will create a secure connection tunneled through an agent to invoke the ticket distribution protocol (similar to `Relay-Ticket-Exchange` – Protocol 1). Agents frustrate a *Sybil attack* [15], where clients join multiple nodes to the system to increase free ticket income, by limiting tickets distributed to each client's IP address.

Distribution Requirements We require several properties as agents distribute tickets: nearly all clients should obtain tickets to remain indistinguishable from relays when spending; the algorithm that assigns clients to agents should not leak the client's identity to prevent an adversary from using a predecessor attack to infer a client from its agents; a client should obtain more tokens by becoming a relay than by cheating to maintain relay incentives; and a client's set of agents should be consistent over roughly the same period as they would if used as regular guards for stability.

Agent Assignment Each client uses Protocol 3 to determine which guard nodes it can use as distribution agents. A fundamental part of the protocol is the *hash–bandwidth test* for a guard $\mathcal{G}$. The test is true if the result of a cryptographic hash is less than the fraction of total guard bandwidth provided by $\mathcal{G}$ ([46] describes secure bandwidth measurements).

A client uses hash–bandwidth tests to walk through the guards while constructing a set of signature chains such that each chain can be verified as a correct chain for the client. After completing Protocol 3, each constructed chain is then used as input to a final round of hash–bandwidth tests (one for each guard): every guard that passes this final test is assigned as a distribution agent for the client.

A client builds signature chains as follows. On line 1, a client initializes a signature chain with its IP address and 0 as the current step in the walk. The client then initiates a pseudo-random walk: any guard $\mathcal{G}$ that passes the hash–bandwidth test is a valid next step, using the previous link in the chain and $\mathcal{G}$'s public key as input to the hash function (line 7). If $\mathcal{G}$ passes the test, it will return a signature to extend the chain and the walk (line 8-9). Although not shown on line 8, the client also sends the previous signature in the chain to prove to the guard that the signature request is valid (and not a waste of resources). Note that each step of a walk may break or fork a signature chain, hence the number of parallel walks performed, depending on the number of passed hash–bandwidth tests. A client will continue extending its signature chains until it has walked *walk_length* steps, or has no next steps for any walk.

Protocol 3 `Agent-Assign`: Clients verifiably compute their assigned ticket distribution agents by creating signature chains.

```
1:  sig_chains = [((clientIP, 0))]
2:  for i = 1 → walk_length do
3:    next_step = [ ]
4:    for chain ∈ sig_chains do
5:      link = last(chain)
6:      for G ∈ guards do
7:        if hash(link | G.pub_key)<bandwidth_frac(G) then
8:          sig = get_sig(G, clientIP, i)
9:          next_step.append(chain + G.pub_key + sig)
10:   sig_chains = next_step
11:   if sig_chains = [ ] then
12:     abort() /* found no valid chains of proper length */
```

If Protocol 3 does not terminate via *abort*(), then each chain in the list of *sig_chains* is a verifiable (but not public) token that attests the correctness of the assignment without leaking the client's IP address. The client uses each constructed chain to find an agent: a guard $\mathcal{A}$ that passes another hash–bandwidth test using the chain and $\mathcal{A}$'s public key as input to the hash function.

If Protocol 3 terminates via *abort*(), then the client does not have any valid agents. However, to control the expected and median number of agents per client, we may add an adjustable parameter λ to the bandwidth fraction of each guard node. Probabilistic bounds show that the probability of having k agents will decrease exponentially in k, making it infeasible for an adversary to gain a large advantage by, e.g. manipulating the public keys of some agents.

Protocol 3 requires that clients compute hashes for every guard, but is advantageous since it does not require reassignment when agents churn and it load-balances distribution tasks among agents. Although Tor directory servers measure bandwidth [51], we require a secure bandwidth measurement technique such as [46]: the bandwidth values listed in the consensus become a security parameter since they determine the outcome of the hash–bandwidth tests, the number of agents assigned to a client, and therefore the total number of free tickets a client may receive.

Longer walks increase security since an adversary must compromise *walk_length* nodes to manipulate a signature chain. We suggest using *walk_length* = 3 so that an adversary has a higher probability of compromising a circuit than compromising a walk: in the random oracle model, and assuming a deterministic signature scheme (like RSA+FDH), predicting (better than random guessing) whether a given guard is a valid agent for a client reduces to producing valid signatures for all *walk_length* signatures in the chain. With a walk length of 3, an adversary would not only need to control the final relay in the chain, but also either the first or second relay to obtain all three signatures. The fraction

of nodes' agents that can be guessed in this way is on the same order of magnitude as the fraction of circuits that can be compromised by end-to-end attacks. We simulated agent assignment and found that using $walk\_length = 3$ and $\lambda = 0$ results in a median of one agent per node (details omitted for space reasons). Note that clients may use unassigned guards, but will be unable to collect free tickets from them.

Since agents limit distribution to unique IP addresses, users behind NAT boxes will compete for handouts and aggregate performance for NAT users will suffer. Note that if IPv6 is universally adopted, the distribution scheme will require modification since each client can generate several IPv6 addresses [37]. We accept an adversary capable of joining multiple IPv4 nodes since it increases Tor's anonymity set.

Agent Collusion Using guards as distribution agents introduces a chance for collusion. An adversary could join a relay to Tor, become a guard, and distribute tickets to a colluding client. To mitigate this problem, the bank will only allow an agent to distribute tickets if that agent has e.g. obtained the "stable" flag in Tor [50]. An agent cannot cheat until it has contributed significant resources.

The bank also limits distribution to each agent. Bandwidth measurements may be used to estimate the number of clients an agent is servicing, and the number of tickets the agent is allowed to distribute. Since agents are also guard nodes and ticket distribution is based on contributed bandwidth, the number of tickets they distribute directly correlates with the number of tickets they earn by relaying traffic. This can be used to bound the advantage agents gain by not honestly distributing tickets to clients.

Suppose agent $\mathcal{A}$ has bandwidth fraction b. Each agent has two non-agent guard nodes, and $\mathcal{A}$ receives $\frac{1}{3}$ of the tickets they spend when they select $\mathcal{A}$ as their entry node to Tor. Each selection occurs with probability b, so $\mathcal{A}$ receives $\frac{2 \cdot b}{3}$ of the tickets just by being a guard. If $\mathcal{A}$ additionally keeps the tickets $\mathcal{A}$ is supposed to distribute, the most tickets $\mathcal{A}$ can receive is $\frac{5 \cdot b}{3}$, about 2.5 times as many tickets. This is the worst-case: tickets re-spent by relays will lower this bound. Future work should consider auditing agents' ticket distribution to detect dishonesty.

Ticket Economy Our ticket distribution strategy continuously introduces new tickets into the system that will eventually be exchanged at the bank. Continuous ticket exchanges impose a bandwidth constraint on the bank (see Section 4.1). Therefore, we must bound the total number of tickets that exist in the system to allow the bank to handle all exchanges.

To bound the total number of tickets in the system, the bank imposes a *ticket tax* on users when exchanging tickets. The tax rate is adjustable based on the bank's bandwidth constraints and estimate of the total number of tickets currently in the system. The bank's estimate considers the number of tickets exchanged during previous exchange intervals (tickets not exchanged expire automatically). In practice, the bank can probabilistically fail each ticket exchange to reach the desired tax rate, but this consumes bandwidth resources for tickets that will be taxed. Alternatively, the bank could reveal random numbers that represent a hash output range during every exchange period, and tickets whose hash value falls in this range can be considered taxed and invalid. Then clients can discover which of their tickets have been taxed without contacting the bank. The anonymity implications involved with taxing and bounding tickets are discussed in Section 4.2.

3.4 Differentiated Service

BRAIDS employs differentiated services and a scheduler based on the proportional differentiation model introduced by Drovolis *et al.* [16, 17, 18, 19]. The model states that performance for each service class (in terms of measurable metrics like queueing delay) should be relatively proportional to parameters configured by the network operator. Let $q_i(t, t+\tau)$ be a performance metric measured during the interval $(t, t+\tau)$ for monitoring time scale τ. The proportional differentiation model creates quality differentiation parameters c_i for each class of service i and introduces constraints such that:

$$\frac{q_i(t, t+\tau)}{q_j(t, t+\tau)} = \frac{c_i}{c_j}$$

where $c_1 < c_2 < \ldots < c_n$. We write the delay ratio between these classes as $c_1 : c_2 : \ldots : c_n$. This means that the performance metric under consideration should always maintain the proportions defined by the quality differentiation parameters, during any monitoring timescale.

We define the performance metric q_i to be the queueing delay of class i; the delay parameters between each class are adjustable. Drovolis *et al.* contribute schedulers that approximate proportional delay differentiation under heavy loads. BRAIDS utilizes the Hybrid Proportional Delay (HPD) scheduler, which is a combination of the Waiting Time Priority (WTP) and the Proportional Average Delay (PAD) schedulers. Each Tor cell is time-stamped upon arrival at the relay and placed in the queue associated with the cell's class of service. When the relay makes a scheduling decision at time t, WTP computes the priority of only the cells at the head of each class i's queue as $p'_i(t) = \frac{w_i(t)}{c_i}$, where $w_i(t)$ is the waiting time of the cell computed using the time-stamp from above. PAD computes class priorities as $p''_i(t) = \frac{a_i(t)}{c_i}$, where $a_i(t)$ is the total average delay incurred by service class i before time t. HPD weights these priorities as $p_i(t) = p'_i(t) \cdot (1 - f) + p''_i(t) \cdot f$, where f is an adjustable fraction. The cell with the highest computed priority $p_i(t)$ is scheduled. In BRAIDS, the HPD scheduler computes at most six priorities for each scheduling decision.

HPD allows us to differentiate performance of paying and non-paying clients by adjusting the c_i parameters. We then divide client traffic into three distinct service classes: (1) *Low-latency* for web browsing clients, (2) *High-throughput* for file sharing clients, and (3) *Normal* for non-paying clients. These classes will be proportionately delayed as low-latency : high-throughput : normal.

Low-latency Service Users who wish to browse the web typically want fast response but not high throughput. Therefore, we schedule low-latency traffic with the highest priority. We rate-limit low-latency traffic for each circuit to prevent users from sending high traffic loads and overwhelming the low-latency class; traffic exceeding a threshold limit over a monitoring timescale will be demoted to the normal class. We suggest a threshold equal to the number of free tickets each user receives during a spending interval (discussed in Section 4.1). Throttling is necessary to prevent high-throughput clients from "abusing the pipe" for web users, which is currently a well-known problem in Tor [32].

High-throughput Service Conversely, clients with high throughput requirements (e.g. BitTorrent users) tolerate higher-latency service. Therefore, we increase scheduling delays relative to the low-latency class but do not throttle

traffic. As a result, high-throughput traffic has a diminishing effect on low-latency traffic.

Normal Service Since not all users will be able or willing to deploy relay services, we do not *require* clients to make payments in order to use BRAIDS. Instead, clients who have expended their free ticket allowance, or choose not to pay for service, receive both the lowest priority and, in turn, the highest scheduling delays.

Differentiating service results in two interesting consequences: it provides incentives to run relays, since users in higher service classes receive lower delay; and it allows for incremental deployability by placing traffic from legacy clients in the normal service class. Note that the extent of the performance gain between service classes depends on the chosen delay parameters.

4. ANALYSIS AND DISCUSSION

4.1 Parameter Selection

Ticket Validity Intervals Recall that ticket validity intervals [-,d_u), [d_u,d_v), and [d_v,d_w) are global uniform intervals in which tickets may be spent and exchanged and are broadcasted by the bank (see Section 3.1). We explore both the frequency and relative time that each interval occurs.

To prevent unspendable ticket periods, tickets that are received in spending interval i are exchanged in spending interval $i+1$ (exchange interval i overlaps spending interval $i+1$). Time in each exchange interval is shared between relay-exchange and client-exchange such that the fraction of time alotted for relay exchange corresponds with the expected fraction of tickets relays possess (which the bank can estimate based on exchanges in previous intervals).

Using the interval strategy just described, the bank will only exchange half of all tickets in the system during every spending interval and users can only spend half of their tickets at one time. Following this approach, tickets received in spending interval i are exchanged in spending interval $i+1$ and spent in spending interval $i+2$. All tickets not exchanged during an exchange interval will expire, so if relays are offline for the duration of an exchange interval, they will lose roughly half of their tickets.

Longer spending and exchange intervals means relays must wait longer to use tickets, but shorter intervals means tickets expire faster. We suggest a compromise of 24 hour spending and exchange intervals, noting that further exploration of exchange intervals is desirable.

Ticket Worth Recall that several cells may be transferred through Tor for each ticket. The number of cells transferred for each ticket has an important impact on the bank's CPU and bandwidth consumption. Since we limit the amount of data users can download for free, higher ticket worth means the bank has to exchange fewer tickets, reducing both CPU and bandwidth requirements. However, users then have fewer tickets overall which reduces the number of independent circuits that can be paid simultaneously. We suggest that users receive 3 tickets every 10 minutes so they may utilize a prioritized circuit at any time. We note that in practice these tickets will likely be freely distributed in batches at a higher time granularity (e.g. every hour).

Cryptographic and Bandwidth Costs Each relay must perform a SHA1 hash and an AES encryption/decryption for each cell it transfers. BRAIDS introduces an additional task – verification of a ticket. We implemented the partially blind signature scheme of Abe *et al.* [1] using GMP [23] for arbitrary precision arithmetic. We measure both the amount of time a bank spends producing a signature, and the amount of time a relay spends verifying a single ticket. We also compute the time to perform the SHA1 and AES operations required by Tor.

		Mean	Median	Std. D.
AMD Athlon (3 GHz)	AES+SHA1	9.139	8.616	2.493
	PBS verify	531.287	530.885	8.342
	PBS bank	413.244	412.069	7.297
Intel Core2 (2.67 GHz)	AES+SHA1	6.226	5.859	1.307
	PBS verify	1496.813	1496.613	10.844
	PBS bank	1193.233	1192.782	9.472

Table 1: Cryptographic time per cell for Tor relays compared with BRAIDS PBS verifications, in microseconds. Also shown is the bank's time per signature.

Table 1 shows the results of our Linux benchmarks on 3 GHz AMD 64 Athlon X2 6000 and 2.67 GHz Intel Core 2 Duo 6750 CPUs. We report the mean, median, and standard deviation of times, in microseconds, for each operation described above. As expected, a signature verification takes significantly longer than AES and SHA1 operations currently performed by relays. However, the value of each ticket can be selected such that a ticket need not be sent for every cell and expensive ticket verification costs can be amortized. An appropriate value would result in a greater cost for AES and SHA1 than for verifications to prevent the signature scheme from becoming a bottleneck. Our benchmarks suggest a single ticket be worth 128 KB of data so that a verification need only be performed for every 256 cells.

Given our Intel benchmarks, a relay performs roughly 666 verifications per second while the bank may perform over 833 signatures per second per processor core. Each relay may therefore upload at a rate of 666 Mbps while streamlining verification procedures, and the bank may sustain an aggregate 833 Mbps of prioritized traffic through Tor. Recall that the bank is offline and may be distributed among multiple machines for additional processing resources.

To compute bandwidth costs, suppose a user receives η free tickets during each spending interval. Not including protocol overhead, which can be minimized by batching ticket exchanges, each ticket exchange consumes 488 bytes of bandwidth (the partially blind signature from Abe et al. [1] requires multiple messages between the client and the bank). In aggregate, the bank distributes $\eta \cdot \mu$ tickets per day, where μ is the total number of users receiving free tickets. Ticket exchanges are taxed such that after ρ spending intervals, $\eta \cdot \mu$ tickets are eliminated from the economy. The total number of tickets in the system is $\eta \cdot \mu \cdot \rho$ in expectation.

Since the spending and exchange intervals overlap, the bank will exchange and produce signatures for $\frac{\eta \cdot \mu \cdot \rho}{2}$ tickets every spending interval. If we assume a spending interval is 24 hours following our interval strategy from above, $\eta = 432$, $\mu = 100,000$, and $\rho = 20$, then the bank must sustain bandwidth loads of 20 Mbps and perform 5,000 signatures per second, within reason of a multi-core CPU with a cryptographic accelerator.

4.2 Security Analysis

To measure the impact of BRAIDS on sender anonymity, we analyze information leakage in terms of an *anonymity probability distribution* [13, 44]. This analysis technique uses information-theoretic entropy [45] as a measure of information contained in a probability distribution. We define a discrete random variable I as a circuit initiator and com-

pute a distribution of all potential initiators as a probability mass function $Pr(I = i) = p_i$ where p_i is the probability that a user i is the circuit initiator given the observations on the system. The entropy H of our distribution is:

$$entropy = H(I) = -\sum_{i=1}^{N} p_i \log_2(p_i)$$

where p_i is the probability for user i taken from the distribution and N is the size of the anonymity set (the set of potential circuit initiators). The maximum entropy in the system H_M is computed as $H_M = \log_2(N)$. The *degree of anonymity* [13] quantifies information leakage and can then be defined as the fraction of total entropy obtained from the given distribution I:

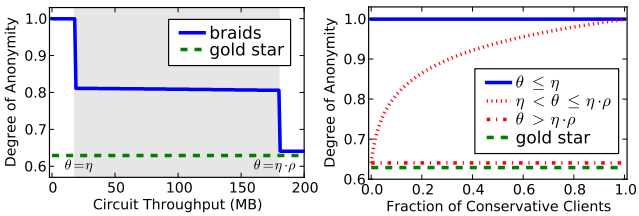
$$degree\ of\ anonymity = \frac{H(I)}{H_M}$$

Distinguishability To determine the effects of distinguishing clients from relays, we first assume that clients will fill one of two roles: a *liberal* client who spends tickets immediately by downloading web pages and a *conservative* client who stores tickets until they can download a large file. Our liberal-conservative client model captures potential BRAIDS spending habits – in practice some clients will consistently spend most of their tickets while others will consistently underspend. We further assume that each relay in the system always has the desired number of tickets for any circuit it initiates to simplify analysis. We note that this is a coarse model as it is difficult to estimate users' spending habits.

While the tax rate ρ allows the bank to remove tickets from the system to keep ticket exchanges within manageable bandwidth bounds, it also potentially reduces anonymity for large downloads. If a user spends more tickets than is possible to collect only from free distribution ($\eta \cdot \rho$), an adversary can determine with high confidence that the circuit initiated from a relay by observing $\theta > \eta \cdot \rho$ tickets spent in a circuit. An adversary may additionally determine *which* relays can afford a given circuit by performing bandwidth measurements, since a relay's ticket income corresponds with the bandwidth it provides. However, in Section 4.1 we have suggested distributing enough free tickets to pay for general web browsing so that the majority of users will not spend over $\eta \cdot \rho$ tickets.

Discussion To analyze our system, we obtain the growth rate of Tor relays from [50]. We estimate the client growth rate by analyzing how the number of client connections to a relay changes over a two month experiment [55]. We apply these rates and the estimated network size of 100,000 clients and 1,500 relays to find the total network size over time. From Section 4.1, each ticket is worth 128 KB of data transfer and we distribute $\eta = 432$ tickets per day. Tickets are taxed such that the system's ticket capacity is $\rho = 20$ cumulative days of tickets. The fraction of conservative clients is $\frac{1}{10}$, except where noted.

Figure 1(a) shows how the fraction of conservative clients may affect the set of potential initiators (if an adversary can guess this fraction) and therefore the degree of anonymity BRAIDS provides. By observing $\theta < \eta$ tickets in a circuit, an adversary is unable to infer information about the circuit initiator since all liberal and conservative clients obtain η tickets during a spending interval. Observing $\theta > \eta \cdot \rho$ tickets means the circuit must have been initiated from a relay. For other observations, the degree of anonymity depends on the number of clients the adversary can eliminate from the potential initiator set.



(a) (b)

Figure 1: Anonymity is highest when the adversary observes fewer than $\theta = \eta$ tickets per circuit and lowest when more than $\theta = \eta \cdot \rho$ are observed. (a) In the shaded area, only $\frac{1}{10}$ of clients are conservative, collecting tickets longer than one spending interval. (b) Anonymity increases with conservative clients that contribute to adversarial uncertainty.

The shaded area in Figure 1(a) represents anonymity when $\frac{1}{10}$ of clients are conservative. This fraction is an estimate: it is difficult to determine how users will spend in practice. We explore the effects of varying the conservative client fraction in Figure 1(b). Since conservative clients represent adversarial uncertainty, we find that having more conservative clients has a positive effect on anonymity. In all cases, anonymity is higher in BRAIDS than the gold star scheme where only the fastest $\frac{7}{8}$ of relays are potential prioritized-traffic initiators. For highest anonymity, clients should spend less than η tickets for prioritized traffic in each spending interval.

5. SIMULATION AND RESULTS

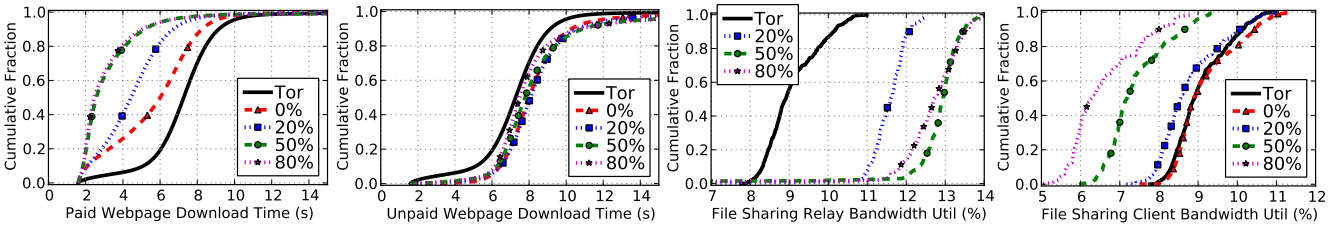
We simulate BRAIDS and Tor to compare performance and illustrate how effective our system is at encouraging users to run relays. Below we describe our simulator, experiments, and results.

5.1 Simulator

We built a discrete-event-based simulator that models the Tor network. Within the first ten minutes of each experiment, all Tor clients start one of the applications described below and begin generating data. Each client builds circuits following Tor's path selection protocol, and refreshes each circuit after ten minutes, building a new one when the next request is made. We now describe our client applications.

Web Clients Each web client (WC) generates traffic by making a top-level page request and waiting for a response from the server. After receiving a response, the WC makes several additional parallel requests for objects embedded in the page (e.g. images). After receiving all embedded object responses, the WC waits for a period of time before downloading another page. We record the time required to download the entire page, including all embedded objects. The period between the initiation of the top level request until the reception of the final embedded object simulates the time required to render an entire page in a user's browser. Distributions for all request and response sizes, the number of embedded objects per page, and the time between page requests are taken from the web traffic study conducted by Hernandez-Campos *et al.* [27].

File Sharing Clients Each file sharing client (FSC) simulates a BitTorrent-like protocol by continuously generating data to five random peers through the Tor network. Every thirty seconds the FSC will replace its slowest connection with a new peer and a new circuit, simulating BitTorrent's



(a) (b) (c) (d)

Figure 2: BRAIDS and Tor simulated performance with a varying percentage of File Sharing Clients converting to File Sharing Relays. Webpage download time for (a) paid, low-latency service, and (b) unpaid, normal service. Bandwidth utilization for (c) file-sharing relays, and (d) file-sharing clients.

"optimistic unchoke" algorithm [6]. Each FSC exchanges blocks by sending a 32 KB request for a 32 KB reply and immediately sending another request upon receiving a reply. We measure the time to exchange each.

File Sharing Relays A file sharing relay (FSR) implements the same algorithm as a FSC with the following deviation: FSRs contribute a fraction of their total upstream bandwidth to Tor while using the remaining bandwidth for their own file transfers. The bandwidth contributed by FSRs supplies them with additional income not received by FSCs.

We simulate every cell generated by each client and sent through the Tor network. Tor nodes schedule outgoing cells using an exponential weighted moving average (EWMA) scheduler [47], while BRAIDS nodes use the HPD scheduler (see Section 3.4). To bootstrap the economy, tickets are distributed to clients and relays at the start of each simulation.

5.2 Experimental Parameters

Our simulated network consists of 19,400 web clients, 300 Tor relays, 2,000 servers, and 600 file sharing nodes. Our web and file sharing nodes are given consumer-class connections of 12 Mbps downstream 1.3 Mbps upstream bandwidth, and 24 Mbps downstream 3.5 Mbps upstream bandwidth, respectively.[3] File sharing relays draw contributed bandwidth amounts from the Tor network consensus [49] repeatedly until obtaining a value below their upstream capacity.[4] Altruistic relays are given symmetric upstream and downstream capacities drawn from the bandwidth distribution reported in the consensus, clipped at 20 MB following standard Tor procedure [52]. Servers are given unlimited bandwidth and we impose no processing delay on any node. Network latency between every hop is set to 100 ms, and we do not account for membership churn or congestion control in our simulator since it will have a similar effect on both Tor and BRAIDS performance. We simulate 60 minutes.

We run BRAIDS and Tor experiments with the above parameters. We run multiple BRAIDS experiments using $1 : 64 : 4096$ as the HPD scheduler's delay parameters corresponding to the service class ratio low-latency : high-throughput : normal, and HPD fraction $f = 0.875$ (see Section 3.4). Since we are interested in the incentives our system provides for running a relay, we vary only the fraction of 600 nodes that are FSCs as opposed to FSRs. This will allow us to determine how a user's performance changes by serving as a relay. The load on the network is unchanged between all experiments. Our simulator closely approximates empirical Tor traffic loads gathered by McCoy *et al.* [32].

[3]ADSL Standard ITU G.992.1 Annex A, ADSL2+ ITU G.992.5 Annex M.

[4]The consensus document was obtained on January 12, 2010 between 18-19:00:00 CST.

5.3 Results

In BRAIDS, the low-latency service class achieves a significant reduction in download time compared with Tor, and download times improve as more FSCs convert to FSRs (Figure 2(a)). Since web browsers transfer small amounts of data in most cases, improvements in download times are noticed even with few new relays. The similarity in download time when 50% and 80% of FSCs change to relays suggests that these nodes have reached a lower bound. We note that the best possible download time is 1.6 seconds, since all web clients must make at least one top-level and one embedded object request, resulting in sixteen 100 ms hops. The normal service class webpage download time is longer than in Tor, and performance slightly declines as more file sharing users move to the high-throughput class since normal data is proportionally delayed sixty-four times as long as high-throughput data (Figure 2(b)). Unpaid traffic performance is best when 80% of the FSCs convert to relays since clients can take advantage of a significant increase in available bandwidth. These results are outstanding – download time for normal web traffic does not unusably degrade from performance achieved in Tor, and running a relay will provide a definite performance boost over those who choose to remain client-only.

BRAIDS FSRs not only receive an improvement in bandwidth utilization over Tor, but can also achieve up to approximately 90% better utilization of their bandwidth compared with BRAIDS FSCs that do not run a relay, even while contributing a fraction of their bandwidth to Tor. Figure 2(c) shows that FSRs performance increases as more nodes convert to relays. However, since the newly available bandwidth is also consumed by WCs, relays realize only incremental improvements as the fraction of converting relays increases. Figure 2(d) shows that as more filesharers convert to relays, performance for FSCs degrades. This happens mostly because a large amount of data from FSRs is receiving priority over data from FSCs, and the newly available bandwidth is being consumed by the low-latency and high-priority service classes. For all conversion rates, FSRs achieve considerably better performance than FSCs.

Overall, our results strongly indicate that BRAIDS users can increase the performance of both interactive and non-interactive traffic by starting a relay and contributing bandwidth to Tor. Therefore, if users want to run BitTorrent or similar file sharing protocols using BRAIDS, they should run a relay to achieve maximum performance. This, in turn, will have a positive impact on the entire network since there will be more bandwidth available for other Tor clients.

6. RELATED WORK

Tor Incentives Research from the community has provided few ideas to produce incentives to run Tor relays [3, 10, 38] in order to utilize recent scalability [34] and performance improvements for Tor [40, 46, 56]. Ngan *et al.* [38] previously proposed a system in which Tor directory servers actively measure the performance of relays and note the "best" relays in the directory with a "gold star". This scheme introduces security vulnerabilities: the anonymity set of relays is significantly reduced since gold star relays can be distinguished from regular relays and the changing membership of the gold star set leads to an *intersection attack* [20, 28, 33, 35].

PAR [3] is another scheme exploring incentive mechanisms for relays. In PAR, a centralized bank issues coins to clients while handling deposits from relays. Relays frequently deposit and verify coins at the bank to limit client double-spending. The need for frequent coin verification introduces a fundamental design problem – a trade-off between double spending detection and anonymity.

Incentives in Other Networks Incentives have been previously proposed for several anonymous and peer-to-peer (P2P) systems. Both Anonymizer.com [4] and the Freedom network [7] introduced commercial anonymity systems based on collecting payments for service. While the latter failed, the former is still in operation and provides a one-hop anonymous proxy based system for paying clients.

Franz *et al.* [22] introduce an incentive technique for mix-networks that divides electronic payments for each mix, but is inefficient since each hop requires communication between the mix and mix provider.. Figueiredo *et al.* [21] also introduce an electronic mix-net anonymity system, but it lacks accountability and robustness. Reiter *et al.* [41] build upon coin ripping [29] to develop a strict fair exchange protocol for mix-nets. However, they require each message recipient to participate in the protocol which does not align with Tor's desire to support arbitrary destinations.

Golle *et al.* [25] discuss incentives for sharing in P2P networks using a game theoretic model. They propose a micro-payment or "points" system, where uploads are rewarded and downloads are penalized, and show that equilibrium is reached by balancing uploads and downloads. BitTorrent [11] uses "tit-for-tat" (TFT) mechanisms to trade pieces of large files among peers. Peers upload and download pieces cooperatively and preferentially from other peers in an attempt to maximize local download efficiency. The Scrivener [36] system uses a credit/debit approach to maintain local histories of other nodes' cooperative behavior which is used to enforce fair bandwidth sharing. Similarly, reputation systems [12, 30, 57, 58] use interaction histories to develop trust levels for other peers which aids in future decisions to cooperate while incentivizing trustworthy behavior. Reputation systems are incompatible with Tor since not all relays are able to bind an interaction to a client.

Acquisti *et al.* [2] discuss incentives to participate in anonymous systems while developing an economic model and arguing a usage fee as an economic incentive mechanism. Cost then is a security objective since it affects the number of users and therefore anonymity provided [5].

7. CONCLUSION

In this paper we introduced BRAIDS as a set of practical mechanisms that encourages users to run Tor relays. We employ completely client-anonymous relay-specific tickets that allow Tor clients and relays to achieve increased performance while preventing the double-spending problem. Relays differentiate service into three classes, allowing them to prioritize traffic types and mitigate the negative effect file sharing users have on Tor, without significantly reducing bandwidth utilization for file sharing clients.

There are several issues that need further investigation. Our uniform ticket validity intervals introduce a trade-off between fast ticket turn-around times and offline relays losing their tickets. The trade-off is due to our imposed ticket tax which is required to bound the bandwidth load on the bank. Tickets you earn will not be usable for 2 days in our current design, and you will lose half of your tickets if you are offline for a day and unable to exchange them.

BRAIDS would benefit from a more robust ticket distribution scheme, since an adversary controlling a fraction of the IP addresses in Tor will likely be able to steal a similar fraction of tickets in each spending interval, and malicious agents receive a greater reward for stealing tickets than behaving honestly. Further, auditing ticket agents would allow us to detect cheaters.

Future work may also consider an exploration of client strategies to enhance our model of client spending habits and improve our anonymity analysis. Users must be aware of their prioritized spending habits since spending a large number of tickets (more than a few hundred MB) will reduce their anonymity and lead to the risk of intersection attacks as in the gold star scheme. Finally, distributing the bank's functionality among users, or a small set of trusted nodes, will reduce bandwidth and CPU limitations, drastically improving anonymity.

Acknowledgments We thank our shepherd, Roger Dingledine, and our anonymous reviewers for many helpful comments. We thank Eugene Vasserman, Eric Chan-Tin, Max Schuchard, Abedelaziz Mohaisen, Paul Syverson, and Prateek Mittal for their helpful discussions, and Zi Lin for his assistance with the PBS implementation. This research was supported by NSF grants CNS-0546162 and CNS-0917154.

8. REFERENCES

[1] M. Abe and T. Okamoto. Provably secure partially blind signatures. In *CRYPTO '00: Proceedings of the 20th International Cryptology Conference on Advances in Cryptology*, pages 271–286, 2000.

[2] A. Acquisti, R. Dingledine, and P. Syverson. On the economics of anonymity. In *Proceedings of the 7th International Conference on Financial Cryptography*, 2003.

[3] E. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S. M. Bellovin. PAR: Payment for anonymous routing. In *PETS '08: Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, pages 219–236, 2008.

[4] The anonymizer. http://anonymizer.com/.

[5] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *IHW '01: Proceedings of the 4th International Workshop on Information Hiding*, pages 245–257, 2001.

[6] The BitTorrent Protocol Specification. http://www.bittorrent.org/beps/bep_0003.html. January 2010.

[7] P. Boucher, A. Shostack, and I. Goldberg. Freedom system 2.0 architecture. White paper, Zero-Knowledge Systems Inc., 2000.

[8] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO '82: Proceedings of Advances in Cryptology*, volume 82, pages 199–203, 1983.

[9] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '88: Proceedings of Advances in Cryptology*, pages 319–327, 1990.

[10] Y. Chen, R. Sion, and B. Carbunar. XPay: practical anonymous payments for Tor routing and other networked services. In *WPES '09: Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society*, pages 41–50, 2009.

[11] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of P2P Systems*, volume 6, 2003.

[12] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216, 2002.

[13] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies*, 2002.

[14] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, 2004.

[15] J. R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, 2002.

[16] C. Dovrolis and P. Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE network*, 13(5):26–34, 1999.

[17] C. Dovrolis and P. Ramanathann. Proportional differentiated services, part II: loss rate differentiation and packet dropping. In *IWQOS'00: Eighth International Workshop on Quality of Service*, pages 53–61, 2000.

[18] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: delay differentiation and packet scheduling. In *SIGCOMM '99: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 109–120, 1999.

[19] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1):12–26, 2002.

[20] N. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on Tor using long paths. In *18th USENIX Security Symposium*, pages 33–50, 2009.

[21] D. R. Figueiredo, J. K. Shapiro, and D. Towsley. Using payments to promote cooperation in anonymity protocols. Technical Report 03-31, University of Massachusetts, 2003.

[22] E. Franz, A. Jerichow, and G. Wicke. A payment scheme for mixes providing anonymity. In *TREC '98: Proceedings of the International IFIP/GI Conference on Trends in Distributed Systems for Electronic Commerce*, pages 94–108, 1998.

[23] The GMP Library. http://gmplib.org/.

[24] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[25] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for sharing in peer-to-peer networks. In *WELCOM'01: Proceedings of the Second International Workshop on Electronic Commerce*, pages 75–87, 2001.

[26] G. Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, December 1968.

[27] F. Hernandez-Campos, K. Jeffay, and F. Smith. Tracking the evolution of web traffic: 1995-2003. In *MASCOTS 2003: The 11th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer Telecommunications Systems*, pages 16–25, 2003.

[28] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? In *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pages 82–91, 2007.

[29] M. Jakobsson. Ripping coins for a fair exchange. In *EUROCRYPT*, pages 220–230, 1995.

[30] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *WWW'03: Proceedings of the 12th International Conference on World Wide Web*, pages 640–651, 2003.

[31] K. Loesing. Measuring the Tor network: Evaluation of client requests to directories. Technical report, Tor Project, 2009.

[32] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker. Shining light in dark places: Understanding the Tor network. In *PETS '08: Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, pages 63–76, 2008.

[33] J. McLachlan and N. Hopper. Don't clog the queue! Circuit clogging and mitigation in P2P anonymity schemes. In *FC'08: The Proceedings of the 12th International Conference on Financial Cryptography and Data Security*, pages 31–46, 2008.

[34] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with Torsk. In *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 590–599, 2009.

[35] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *SP '05: Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 183–195, 2005.

[36] A. Nandi, T.-W. J. Ngan, A. Singh, P. Druschel, and D. S. Wallach. Scrivener: providing incentives in cooperative content distribution systems. In *Middleware'05: Proceedings of the ACM/IFIP/USENIX 2005 International Conference on Middleware*, pages 270–291, 2005.

[37] T. Narten and R. Draves. Privacy extensions for stateless address autoconfiguration in ipv6. http://tools.ietf.org/html//rfc3041, 2001.

[38] T.-W. J. Ngan, R. Dingledine, and D. S. Wallach. Building incentives into Tor. In *FC'10: The Proceedings of Financial Cryptography*, 2010.

[39] I. Osipkov, E. Y. Vasserman, N. Hopper, and Y. Kim. Combating double-spending using cooperative P2P systems. In *ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems*, page 41, 2007.

[40] J. Reardon and I. Goldberg. Improving Tor using a TCP-over-DTLS tunnel. In *Proceedings of the 18th USENIX Security Symposium*, 2009.

[41] M. Reiter, X. Wang, and M. Wright. Building reliable mix networks with fair exchange. In *ACNS'05: The Proceedings of the Third International Conference on Applied Cryptography and Network Security*, pages 378–392, 2005.

[42] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[43] R. L. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. In *Proceedings of the International Workshop on Security Protocols*, pages 69–87, 1997.

[44] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, pages 41–53, 2002.

[45] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[46] R. Snader and N. Borisov. A tune-up for Tor: Improving security and performance in the Tor network. In *NDSS'08: Proceedings of the Network and Distributed Security Symposium*, 2008.

[47] C. Tang and I. Goldberg. An improved algorithm for Tor circuit scheduling. Technical Report CACR 2010-06, University of Waterloo, 2010.

[48] Tor partially blocked in China. Tor Project. https://blog.torproject.org/blog/tor-partially-blocked-china. October 2009.

[49] Tor Directory Protocol, Version 3. Tor Project. http://gitweb.torproject.org/tor.git?a=blob_plain;hb=HEAD;f=doc/spec/dir-spec.txt. January 2010.

[50] Relay Flags. Tor Project. http://git.torproject.org/checkout/metrics/master/out/dirarch/relayflags.csv. November 2009.

[51] Computing Bandwidth Adjustments. Tor Project. http://gitweb.torproject.org/tor.git?a=blob_plain;hb=HEAD;f=doc/spec/proposals/161-computing-bandwidth-adjustments.txt. November 2009.

[52] Tor Path Specification. Tor Project. http://gitweb.torproject.org/tor.git?a=blob_plain;hb=HEAD;f=doc/spec/path-spec.txt. January 2010.

[53] The Tor Project. https://www.torproject.org/.

[54] Y. Tsiounis. Efficient electronic cash: new notions and techniques. *College of Computer Science*, 1997.

[55] E. Vasserman, R. Jansen, J. Tyra, N. Hopper, and Y. Kim. Membership-concealing overlay networks. In *CCS'09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 390–399, 2009.

[56] C. Viecco. UDP-OR: A fair onion transport design. In *HOTPETS'08: Proceedings of Hot Topics in Privacy Enhancing Technologies*, 2008.

[57] L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16:843–857, 2004.

[58] R. Zhou and K. Hwang. PowerTrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460–473, 2007.