Collusion-resilient Credit-based Reputations for Peer-to-peer Content Distribution

Nguyen Tran Jinyang Li Lakshminarayanan Subramanian
New York University

1 Introduction

With growing demand for high-quality multimedia content, content providers face enormous pressure to scale the serving capacity. Peer-to-peer content distribution is a natural low cost option to scale system capacity. In a P2P CDN model, content providers serve content using a small number of "official" seeder nodes and rely on participating users to act as individual seeders for others in the system. Although P2P CDNs have the potential to drastically reduce the required serving capacity of official seeders, they must address the challenge of incentivizing users to stay online in the P2P network and act as seeders.

Unfortunately, the incentive mechanisms provided by BitTorrent [4] and its many variants [7, 13] are insufficient. BitTorrent only incentivizes peers that are *actively* downloading the same file to upload to each other. Once a user completes a download, he has no incentive to act as a seeder. In practice, most content distribution sites (e.g. YouTube, NetFlix VoD) consist of a large number of files that attract many users but not enough of them to ensure multiple simultaneous downloaders at all times. An ideal incentive mechanism should require users to contribute to the P2P CDN even after completing their downloads. This is also referred to as the *seeder promotion* problem [16]).

The importance of seeder promotion can be witnessed in private BitTorrent communities. These private communities enforce sharing ratios among its members. For example, in TorrentLeech, each peer must serve as a seeder for 24 hours and its uploaded amount needs to be at least 0.4 times of its downloaded amount. These rules incentivize peers to become seeders. As a result, the download speeds and the availability of content in private BitTorrent communities are significantly higher. A recent measurement study [10] shows that private communities achieve 3-5× the median download speed of public communities and possess 10-50× more seeders than those in public communities. However, private BitTorrent communities require trustworthy participants: they rely on peers to self-report their upload/download volumes and can be easily manipulated by selfish nodes [11].

To motivate selfish peers to become seeders, a P2P CDN needs to have the desirable property that the more a peer contributes (in terms of its uploads) to the system, the better the service (in terms of download speed) it gets. In this paper, we propose Credo, a *credit-based reputation* system that achieves this property and is robust to various attacks. In Credo, peers issue credit tokens to uploaders when downloading data from them and collect credit tokens by uploading to others. Each credit token is explicitly associated with the identity of the original credit issuer. Nodes can either transfer credits received from other peers or issue new original credit tokens. Unlike currency systems [12, 16, 21] which suffer from the bankruptcy problem, Credo allows each node to mint its own credits, thus ensuring no honest node ever goes bankrupt.

Credo uses the credits to compute a *reputation score* which dictates how a peer allocates its upload bandwidth to downloaders. The naive way of counting the number of credits as a node's reputation score is susceptible to the collusion attack where a set of colluders swap credits among themselves without doing actual uploads. Credo employs two techniques to defend against such an attack. First, a node's reputation is computed by its *credit diversity* which measures the number of distinct credit issuers among a node's collected credits. Doing so can effectively limit the reputation score of k colluders to at most k. Second, Credo explicitly *models the distribution of the amount of self-issued credits* by all nodes and filters a node's collection of credits according to the measured distribution. This prevents colluders from launching sustained attacks by introducing Sybil identities that generate arbitrary number of credits. Using the credit diversity and distribution modeling mechanism, Credo can effectively handle collusion attacks.

We believe that Credo is an attack-resilient P2P CDN system that addresses many of the limitations of existing currency and reputation based P2P CDN solutions and also provides a better incentive model to address the seeder promotion problem.

2 Seeder promotion problem

We model the seeder promotion problem in a P2P CDN as follows. We assume all peers are selfish. The utility of each peer is characterized by its average download speed and the goal of each peer is to employ a strategy that maximizes its download speed. It is worth pointing out that a selfish peer is *not* necessarily interested in minimizing its upload cost: each user has a different threshold for acceptable upload cost. In order to motivate peers to serve as seeders, a CDN must be "fair". We use an intuitive notion of fairness: *the more a peer contributes to the system (i.e. uploads) relative to its consumption (downloads), the better average download speed it experiences when competing with other downloaders.* Our model of selfish peers is similar to that proposed in [7]. However, the model in [7] is used to study the incentives of BitTorrent while our model aims to capture the seeder promotion problem.

Our fairness definition is more flexible than enforcing a strict sharing ratio as done in private BitTorrent communities. With a strict sharing ratio, a highly provisioned peer has no incentives to upload more than what is necessary to meet its sharing ratio. Worse yet, a peer unable to meet the sharing ratio requirement for various non-selfish reasons (e.g. it is seeding unpopular files or has small upload capacity compared to its download capacity) risks getting expelled from the system. We also point out that our fairness notion implicitly captures "market price" variations according to supply and demand: when supply far exceeds demand as exemplified by few or no concurrent downloaders for any given file, a peer does not need much contribution in order to obtain a good download speed. On the contrary, when demand far exceeds supply, a peer competing with many concurrent downloaders needs to have contributed much more to obtain a good download speed.

Our fairness notion maps naturally to a reputation system where each peer's reputation score reflects its net contribution (i.e. its uploads minus its downloads) and each peer allocates its upload capacity to active downloaders according to their reputation scores. However, this reputation based approach faces two practical challenges: (a) how to capture a peer's net contribution? (b) how to defend against attacks on the reputation system itself?

The goal of our work is design a reputation system for P2P CDNs that addresses both these challenges.

2.1 Related work

We survey existing proposals of P2P reputation and currency systems and discuss why they do not completely address the seeder promotion problem.

The primary problem with existing reputation systems [5, 6, 9, 14] is that a peer's reputation score does not accurately reflect its net contribution. Existing reputation systems calculate the reputation of a node based on the interaction graph between nodes – where each graph edge exists between a pair of nodes that have uploaded or downloaded to each other. EigenTrust [6] uses the PageRank-style [1] propagation algorithm on the interaction graph. OneHop reputation [14] and multi-level tit-for-tat [9] restricts the PageRank-style propagation to one or a few hops in the graph. Others have also proposed to use the max-flow computation on the graph [2,5]. With graph-based reputation calculation, a peer's reputation score is heavily influenced by its topological position in the graph. Thus, a strategic peer can gain unfair advantage by selectively contributing to certain peers. As a result, graph-based reputations do not satisfy the desired fairness property.

Currency systems such as Dandelion [16], BitStore [15], PACE [3], Antfarm [12] and others [20, 21] incentivize peers to upload to others in exchange for tokens that entitle them to future downloads. Currency system enforces the strict "download as much as you upload" policy where a node with no currency is not allowed to download. Compared to our notion of fairness, the policy enforced by currency systems is less desirable: since all peers achieve the same download speed as long as they have non-zero currency tokens, a peer has no incentives to contribute more than what is necessary to satisfy its own demand. Moreover, currency systems typically rely on a central party to mint currency tokens and thus face the daunting challenge of maintaining liquidity according to current demand and hoarding levels at all times. Otherwise, many peers can go bankrupt and be unable to download due to the lack of currency tokens.

3 Credo design

The Credo network consists of a large collection of peer nodes as well as a central server trusted by all peers. The central server admits new nodes into the system in a Sybil-resilient fashion: we can either require users to present a strong type of identity (e.g. credit card numbers, cellphone numbers) or use social-network based admission control [18,19,23,24]. It is essential to restrict the number of admitted Sybil identities per user. Otherwise, a collusion group could have an arbitrary number of (Sybil) identities. We assume that the admission control algorithm successfully limits each adversary to a few (s) Sybil identities. Each newly admitted identity obtains its certified public/private key pair from the central server.

Credo uses the idea of *credit-based reputations* to better incentivize nodes to act as seeders and also be resilient to a wide range of attacks. At a high-level, Credo keeps track of upload and download activities of peers using "credits", which are signed tokens confirming the transfer of a data chunk in the system. Each credit is associated with the original issuer of the credit and the chain of nodes that the credit has traversed during data chunk transfers. Each node is free to mint its own credits with monotonically increasing local sequence numbers. Credo also supports mechanisms to expire credits to prevent credit hoarding.

When a seeder is to upload a data chunk to a leecher, it picks a credit from among the leecher's credit pool and requests the leecher to transfer the chosen credit in exchange for the upload. If the leecher's credit pool is empty, it needs to issue a new credit and give it to the seeder. Each credit is a token originally signed by the issuer. Furthermore, when A gives a credit to B, A appends B's identity to the credit and signs the resulting credit chain with its own key. Having an explicit signature chain allows us to detect potential double-spenders. We note that when a pair of peers both have data chunks of interest to each other, they can still employ the BitTorrent-like protocols to exchange data chunks without any credit transfer.

To incentivize seeders, Credo must satisfy the fairness

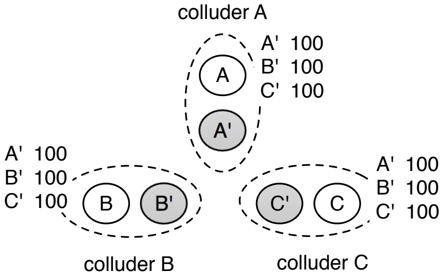

Figure 1: Nodes A,B,C collude by exchanging credits issued by their respective Sybil identities (A',B',C'). Each colluder's credit diversity is increased to 3. Colluders may try to sustain a credit diversity of 3 for their downloads by having Sybil identities continuously issue new credits to replace used ones.

property outlined in § 2 to give peers with more net contribution better download speeds. In a credit-based reputation mechanism, the reputation score of a peer is calculated based on the set of credits that it has collected from others (called *credit pool*). In the absence of collusion, one can measure a peer's net contribution as the *difference* between the number of credits in a peer's credit pool and the number of self-issued credits by the peer. In § 4, we will describe the ideas of *credit diversity* and *modeling good behavior* which describe how this simple reputation metric can be modified to handle collusion attacks.

To calculate a leecher's reputation score, the seeder asks each leecher to transfer its credit pool and report the number of its self-issued credits. Credo can detect nodes that lie about the number of self-issued credits (see § 5); misbehaving nodes can be expelled from the system by the central server.

When uploading data, a peer (seeder) uses the reputation scores of requesting peers (leechers) to provide differentiated service across competing leechers. Each seeder dedicates a fixed number of upload slots and picks leechers with the top reputations among all requesters to serve. After a leecher X is chosen to be served, the seeder verifies that X's credit pool size is indeed larger than that of the highest unchosen leecher by transferring appropriate number of credits from X. Next, we describe how Credo computes the revised credit-based reputation score to handle collusion attacks.

4 Credo's defense against collusion

The biggest challenge facing Credo is to defend against the collusion attack where a set of colluding peers exchange credits to boost each other's reputation without performing any actual uploads. Unlike other types of attacks such as double-spending, colluders leave no provable evidence for their misbehavior. Credo employs two techniques to limit the effectiveness of collusion: (a) credit diversity, (b) modeling good behavior.

4.1 Credit diversity

Recall that our basic design uses the credit pool size for calculating a node's reputation score. Such a scheme is extremely vulnerable to collusion since an adversarial node can easily boost its reputation by having its few Sybil identities to issue a large number of credits. Credo deters this attack by measuring the credit diversity of a node's credit pool as the number of distinct issuers among a node's credit pool. Credit diversity differs from credit quantity in that each unique issuer is counted only once; hence, a seeder node is incentivized to increase its credit diversity instead of credit quantity. Let $\mathcal{C}$ be a node's credit pool, $d(\mathcal{C})$ be the set of distinct issuers in the credit pool and x denote the number of self-issued credits. A node's reputation is calculated as follows:

$$rep = |d(\mathcal{C})| - \rho \cdot x \tag{1}$$

In Equation 1, ρ is an constant parameter greater than 1 so that a node prefers spending credits collected from others over issuing new credits; this technique avoids overhead of the system caused by redundant credits. In this augmented design, if an adversarial node has only a few Sybil identities, it can only increase its credit diversity slightly without uploading data to honest nodes. On the other hand, honest nodes can increase its credit diversity quickly by collecting diverse credits from highly reputed downloaders which have big and diverse credit pools, or by uploading data to many different nodes. Furthermore, even if a set of k adversaries, each bring in s Sybil identities and collude by exchanging credits among themselves, the maximum reputation of each colluding adversary is at most $k \cdot s$. As seen in Figure 1, each adversarial node (A, B, C) can achieve a reputation score of at most 3 with three Sybil identities (A',B',C') despite the fact that each Sybil identity has issued 100 credits.

4.2 Modeling good behavior

Although using credit diversity can bound the maximum reputation score of a set of colluding adversarial nodes, each colluder can still retain its maximum reputation no matter how much data it downloads. That is because every Sybil node can issue arbitrarily many new credits to replenish the credit pool of an adversarial node. For example, in Figure 1, each adversarial node has 100 credits from every Sybil identity. Suppose node A has used up all 100 credits from C' to download 100 units of data, it can always request for more credits from C' to maintain the reputation score of 3 in future downloads.

To mitigate such sustained collusion attacks, Credo models the typical behavior of honest nodes. More concretely, let Z be the random variable of the number of self-issued credits of the issuer of a randomly chosen credit. We measure the distribution of Z for the overall system and use it to filter a node's credit pool to obtain a subset of credits, $\mathcal{C}' \subset \mathcal{C}$, so that the distribution of the number of self-issued credits by issuers in $\mathcal{C}'$ approximates the Z-distribution. We augment Equation 1 to use the filtered pool ($\mathcal{C}'$) for calculating credit diversity. For an honest node whose credit pool consists of randomly chosen credits, this filtering step has little effect. On the other hand, filtering significantly limits attackers' ability to carry on sustained attacks. In Appendix A, we prove that the maximum downloads an adversarial node can perform with maximum reputation is $s \cdot \gamma \cdot \bar{x}$, where s is the number of Sybil identities controlled by each adversarial node, $\bar{x}$ is the average number of self-issued credits of honest issuers and γ is a small constant. Intuitively, this result says that an Sybil node can issue only $\gamma \cdot \bar{x}$ credits in average and send them to the adversary node; this amount of credits is close to the amount of credit issued by an honest node in the system ($\bar{x}$).

The central server is in charge of computing and representing the Z-distribution periodically every τ time unit, where τ the expiration time for credits. Let X be the random variable of the number of credits issued by a random node that has issued non-zero credits. The central server can easily measure the distribution of X by sampling a random subset of nodes in the system: it picks a random node ID and asks for one of the node's witnesses for number of credits issued by that node in the last τ time period. There is the risk that adversarial nodes can skew the distribution of X toward the high end by using a few Sybil identities to issue a huge number of credits. Therefore, we use the truncated distribution of X that excludes a small fraction (δ) of issuers that have issued the most credits. As a result, as long as the colluding set of Sybil identities do not exceed δ fraction of all nodes, they cannot affect the measured distribution X.

We model an honest node's credit pool as a set of randomly chosen credits in the system. Let Z be the random variable of the number of credits issued by the issuer of a randomly chosen credit. We expect the distribution of the number of credits minted by those issuers seen in an honest node's credit pool to resemble the Z-distribution. There is a clear relationship between distribution of Z and that of X, namely:

$$Pr(Z = x) \quad = \quad \frac{Pr(X = x)x}{E(X)}$$

The central party represents Z-distribution using a set of probability density bounds that correspond to m bins, as shown in Figure 2. Let the range of the i-th bin be $[b_i, b_{i+1})$. The ranges of the bins are chosen to so that the size of successive bins increases exponentially, i.e. $\frac{b_{i+1}}{b_i} = \gamma$ where γ is a small constant bigger than 1. Our proof of Credo's collusion resilience relies on the choice of γ (Appendix A). The probability density of the i-th bin is calculated as $Pr(b_i \leq Z < b_{i+1})$ and its lower bound (p_i) can be derived from the distribution of X as follows:

$$\begin{aligned} p_i &= \frac{Pr(b_i \leq X < b_{i+1}) \cdot b_i}{E(X)} \\ &\leq \frac{\sum_{b_i \leq x < b_{i+1}} Pr(X = x) \cdot x}{E(X)} \\ &\leq Pr(b_i \leq Z < b_{i+1}) \end{aligned}$$

The set of lower bounds (p_i) will be used to check if the distribution Z' of a given set of credit $\mathcal{C}'$ is close to Z. A node periodically asks the central server for the set of lower bounds every τ time unit. Given a set of credit $\mathcal{C}'$, a seeder can directly test if Z' satisfy the set of lower bounds since each credit embed the number of credits its issuer issued in the last τ time unit. Finding the subset $\mathcal{C}'$ of the original credit pool $\mathcal{C}$ is slightly harder. A leecher can do exhausted search to find the optimal $\mathcal{C}'$ for its own interest. It can also perform a greedy heuristic as follow.

Given a credit pool $\mathcal{C}$, we first classify each credit into the $i - th$ bin if the number of credits (z) minted by the issuer of that credit is in the range $[b_i, b_{i+1})$ as shown in Figure 3. Let c_i be the number of credits classified to the i-th bin. In order for shaping the filtered pool $\mathcal{C}'$ according to the measured probability density lower bounds, we must ensure that:

$$\frac{c_i}{|\mathcal{C}'|p_i} \geq 1 \qquad \forall i \tag{2}$$

We start with the original credit pool $\mathcal{C}$ and check if it passes the test in Equation 2. If the test fails and the i-th bin having the highest value of $\frac{c_i}{|\mathcal{C}'|p_i}$, we remove one credit from the i-th bin. If there are multiple credits with the same issuer from that bin, we choose to remove one of them. Otherwise, a randomly chosen credit is removed. We repeat this process until the test passes or no more credits are left. The remaining credits form $\mathcal{C}'$.

4.3 Discussion

The techniques presented in §4.1 and §4.2 mitigate collusion but also affect honest nodes in two ways. First, if a group of honest nodes predominantly upload data to each other as opposed to those outside the group, their reputation scores will be lower than their net contribution. Thus, by penalizing certain collusive behavior, Credo also penalizes (hopefully rare) honest behavior that resembles collusion. When considering a long period of time, we expect most honest nodes to be able to upload data to a diverse set of other nodes. Second, an honest seeder may not always have incentive to serve the leecher with the best reputation. For example, if a seeder has already served leecher X with $rep_X < 0$, it will prefer serving Y over X even though $rep_Y < rep_X$. This is because the seeder can improve its diversity with a new credit issued by Y instead of another credit issued by X[1]. We can mitigate

[1] The seeder will continue to prefer X if X has a non-zero credit pool since the seeder is free to choose credits with different issuers from X's

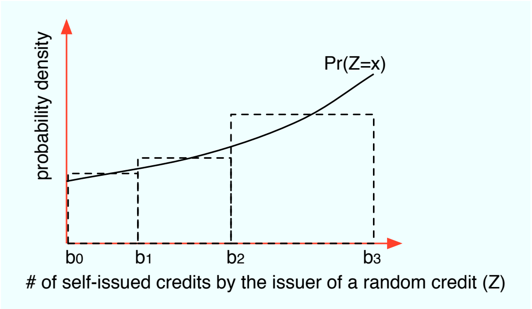

Figure 2: Credo models the behavior of honest nodes with Z-distribution (Z is the random variable of the number of self-issued credits by the issuer of a randomly chosen credit). Credo represents Z-distribution using m bins.

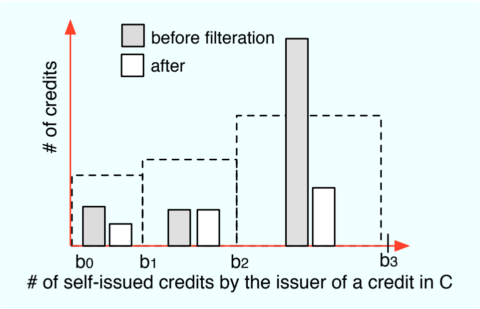

Figure 3: Credo limits sustained collusion attack using the p_i test. In the example, the grey bars correspond to $\mathcal{C}$ of an adversarial node. The white bars correspond to the resulting $\mathcal{C}'$ that fits Z-distribution (dotted lines).

this problem by counting $\delta > 1$ credits from each issuer when measuring credit diversity; the tradeoff is that the maximum reputation score of an adversary also increases as a result. Once a seeder serves δ chunks to a leecher with a negative reputation, it will prefer uploading another node. However, doing so is actually better for the global "good" as there will be more bilateral exchange opportunities among leechers.

5 Detecting misbehavior

Apart from collusion, Credo also faces other attacks such as double-spending and under-reporting of self-issued credits. Unlike collusion, these attacks leave provable evidence for a node's misbehavior due to the various public signatures required by the protocol [8]. Credo uses a lazy auditing protocol to detect misbehavior. We give a few example detections and omit the rest of the details due to space constraints. To detect double-spenders, each node periodically reports a credit it has gotten from peer X to X's witness nodes (Credo uses a DHT [17] to decide X's witness nodes). When X double-spends a credit to both Y and Z, the witness nodes of X will receive two signature chains that show X has signed the transfer of the same credit to both Y and Z, thereby provably detecting X's misbehavior. Similarly, we can also detect if a leecher has under-reported the number of its self-issued credits or if a peer has issued the same credit multiple times. We use the protocol in [8] to catch deviant nodes during the process of exchanging chunks for credits.

6 Preliminary Evaluation

In this section, we demonstrate using simulations how Credo's reputation mechanism successfully reflects a node's net contribution and is resilient to collusion. Using a deployment on PlanetLab on 210 nodes, we also show how Credo improves system performance when nodes are incentivized to act as seeders.

6.1 Simulations

We simulate a peer-to-peer content distribution network of $n = 3000$ nodes. In our simulations, each node has a upload limit of $200KB$ per second. A node divides its upload capacity to 4 upload slots of $50KB$ per second each. Download capacity of a node is 5 times the upload capacity, i.e. 20 download slots. We control the actual upload contribution of a node by a parameter: *willingness*. When a node (seeder) has a free upload slot, it decides to upload to some node (leecher) with a probability proportional to its willingness. We set the willingness of nodes in our simulation to follow the distribution of upload capacity as measured in [13]. We inject new files of $100MB$ to the system sequentially: a new file is injected when all nodes that want the previous file have finished downloading it. Not every node wants every file; instead, each node has a particular demand. When we inject a new file, we choose 300 nodes to download the file. The probability that a node is chosen is proportional to its demand. We model the demand of nodes follows the demand distribution in the Maze file sharing system [22]. We also choose 10 random other nodes as the initial seeders. We choose 30 nodes to be adversaries. Each adversary node controls 3 Sybil nodes. They collude with each other to form a collusion size of 90, i.e. $< 5\%$ of the system. In every $\tau = 3$ days interval, the Sybil nodes issue credits with the optimal amount such that the adversaries' credit pool $\mathcal{C}$ pass the Z-distribution test, i.e. achieving the bound in Observation A.1. The credits are divided equally to the adversary nodes. Adversaries use those credits to download files. We vary the the number of files the 30 adversaries want to download in τ interval in different runs of the simulations. Our simulation stops after simulating the peer-to-peer system for 1 year.

Figure 4 shows nodes' average reputation scores as a function of their net contribution. Our analysis shows that Credo reputation score reflects a node net contribution correctly if nodes follow the protocol. The slope of the curve for the negative net contribution region is bigger

credit pool to improve its diversity.

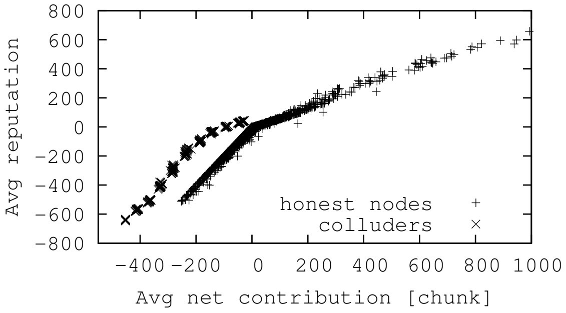

Figure 4: Reputation of colluding adversaries and honest nodes as a function of net contribution.

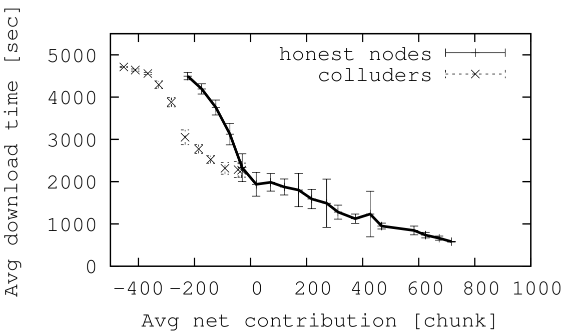

Figure 5: Download time of colluding adversaries and honest nodes as a function of net contribution.

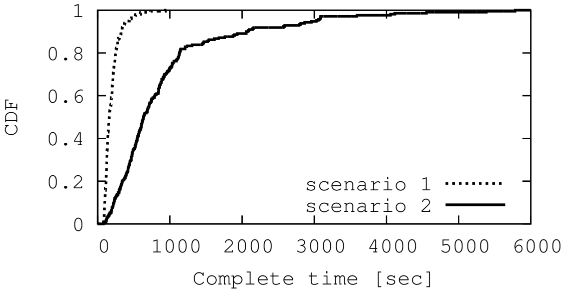

Figure 6: Cumulative distribution of download time two scenario: 1) nodes stay online and continue upload to others, 2) nodes go off line immediately after getting a file.

than that for the positive net contribution region because we set $\rho = 2$. A node with negative net contribution usually has empty credit pool. Downloading one data chunk decreases its net contribution by 1, and decreases its reputation by 2 since it has to issue new credit. On the other hand, nodes with a positive net contribution can spend one credit in their credit pool to download a chunk resulting in decreasing both net contribution and reputation by 1.

In Figure 4, we also plot the reputation of the adversary nodes in different simulation runs with the different number of files they want to download. Because the adversary nodes never upload to other nodes, their net contributions are always negative. Their maximum reputation score is 90 because there are 90 Sybil nodes in this simulation. The Sybils issue 139 credits in average in the optimal strategy, making each adversary hold 417 credits. When an adversary node downloads more than 4 files in one τ interval, it issues new credits and the reputation drops below 0.

In Figure 5, we plot average download time as a function of the net contribution for both adversary nodes as well as honest nodes. As expected, honest nodes with higher net contribution get better download times. This creates incentive for nodes to contribute more. Adversaries can get better download times from collusion but the download times cannot be better than honest nodes with a net contribution of 90.

6.2 PlanetLab deployment

We have a preliminary implementation that integrates Credo credit-based reputation system into the popular Azureus BitTorrent client. We keep the bilateral exchange protocol between nodes intact and only modify the protocol between seeder and leecher: exchanging data chunk for credit, picking the top reputation leecher to unchoke. We have deployed the system on PlanetLab on 210 nodes.

To examine the real performance benefits when nodes are incentivized to contribute, we compare two scenarios. In the first scenario, nodes run our modification of Azureus BitTorrent that integrates the Credo protocol. Nodes are incentivized to serve other nodes in order to gain reputation after downloading a file. In the second scenario, nodes run the original Azureus BitTorrent. They go off line immediately after their download completes.

We inject a 25 MB file to one seeder at the beginning and other nodes arrive to download the file one at a time every 15s. We set the application limit throughput using the distribution in [13] and the download throughput limit is 5 times the upload limit. In the Credo version of Azureus, a leecher pays one credit to a seeder after downloading 250 KB from the seeder. Figure 6 plots the cumulative distribution of complete download time in both scenario. We observe that both the average and the median download time improve significantly when nodes are incentivized to stay online using Credo in scenario 1. The average download time drops from 935 seconds (scenario 2) to 347 seconds (scenario 1). The median download time also drops from 638 seconds to 172 seconds. This result shows that the aggregate capacity of the system improves 2.7 times when nodes are incentivized to contribute.

7 Conclusion

This paper describes the design of Credo, a robust credit-based reputation system that address the seeder promotion problem in P2P CDNs and is robust in the face of collusion attacks and Sybil attacks. Credo addresses the limitations of existing reputation and currency-based systems and Credo's credit-based reputation mechanism better reflects a nodes' true net contribution.

References

[1] Brin, S., and Page, L. The anatomy of a large-scale hypertextual web search engine. In *WWW* (1998).

[2] Cheng, A., and Friedman, E. Sybilproof reputation mechanisms. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems* (2005).

[3] Christina Aperjis, M. J. F., and Johari, R. Peer-assisted content distribution with prices. In *CoNext'08: Proc. ACM SIGCOMM Conference on emerging Networking EXperiments* (2008).

[4] Cohen, B. Incentives build robustness in bittorrent. In *Economics of Peer-to-Peer Systems* (2003).

[5] Feldman, M., Lai, K., Stoica, I., and Chuang, J. Robust incentive techniques for peer-to-peer networks. In *Proceedings of Electric Commerce* (2004).

[6] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web* (New York, NY, USA, 2003), ACM, pp. 640–651.

[7] Levin, D., LaCurts, K., Spring, N., and Bhattacharjee, B. Bittorrent is an auction: analyzing and improving bittorrent's incentives. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication* (New York, NY, USA, 2008), ACM, pp. 243–254.

[8] Li, H., Clement, A., Wong, E., Napper, J., Roy, I., Alvisi, L., and Dahlin, M. Bar gossip. In *Proceedings of the 7th Operating System Design and Implementation (OSDI)* (2006).

[9] Lian, Q., Peng, Y., Yang, M., Zhang, Z., Dai, Y., and Li, X. Robust incentives via multi-level tit-for-tat. In *Proceedings of IPTPS* (2006).

[10] Meulpolder, J., Acunto, L. D., Capota, M., Wojciechowski, M., Pouwelse, J., Epema, D., and Sips, H. Public and private bittorrent communities: A measurement study. In *IPTPS* (2010).

[11] Moya, J. Bram cohen: Private sites to blame for ratio cheating. `http://www.zeropaid.com/news/7728/bram_cohen_private_sites_to_blame_for_ratio_cheating/`.

[12] Peterson, R. S., and Sirer, E. G. Antfarm: Efficient content distribution with managed swarms. In *Proceedings of the 6th conference on Networked Systems Design & Implementation (NSDI)* (Berkeley, CA, USA, 2009), USENIX Association, pp. 1–1.

[13] Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., and Venkataramani, A. Do incentives build robustness in bittorrent? In *NSDI* (2007).

[14] Piatek, M., Isdal, T., Krishnamurthy, A., and Anderson, T. One hop reputations for peer to peer file sharing workloads. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2008), USENIX Association, pp. 1–14.

[15] Ramachandran, A., das Sarma, A., and Feamster, N. Bitstore: An incentive-compatible solution for blocked downloads in bittorrent. In *Proc. Joint Workshop on The Economics of Networked Systems and Incentive-Based Computing (NetEcon)* (2007).

[16] Sirivianos, M., Park, J. H., Yang, X., and Jarecki, S. Dandelion: Cooperative content distribution with robust incentives. In *USENIX Annual Technical Conference* (2007).

[17] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking 11*, 1 (Feb. 2003), 149–160.

[18] Tran, N., Li, J., Subramanian, L., and Chow, S. S. Brief announcement: Improving social-network-based sybil-resilient node admission control. In *Symposium on Principles of Distributed Computing* (2010).

[19] Tran, N., Min, B., Li, J., and Submaranian, L. Sybil-resilient online content voting. In *Proceedings of 6th USENIX NSDI* (Apr. 2009).

[20] Vishnumurthy, V., Chandrakumar, S., and Sirer, E. Karma: A secure economic framework for peer-to-peer resource sharing. In *P2P-ECON* (2003).

[21] Yang, B., and Garcia-Molina, H. Ppay: micropayments for peer-to-peer systems. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security* (New York, NY, USA, 2003), ACM, pp. 300–310.

[22] Yang, M., Chen, H., Zhao, B. Y., Dai, Y., and Zhang, Z. Deployment of a large-scale peer-to-peer social network. In *USENIX WORLDS* (2004).

[23] Yu, H., Gibbons, P., Kaminsky, M., and Xiao, F. Sybillimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy* (2008).

[24] Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. Sybilguard: defending against sybil attacks via social networks. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications* (2006).

A Defending against collusion

In this section, we present analysis results to quantify how Credo limits sustained collusion attacks.

Colluding adversarial nodes exchange credits issued by their Sybil identities (see example in Figure 1). Each adversarial node can have s Sybil identities where $s = O(1)$. Since colluders are self-interested adversaries, they aim to fairly divide the credits among themselves so that each adversarial node is benefitted equally from the collusion. For the simplicity of discussion, we assume adversarial nodes do not contribute any uploads to the system. Furthermore, we assume the set of colluding adversaries is smaller than the threshold $\delta * n$, where δ is the parameter used for measuring the truncated distribution of X.

Observation A.1 *Suppose there are k self-interested colluding adversarial nodes, each with s Sybil identities. Credo limits the maximum reputation of an adversarial node to be $k \cdot s$. More importantly, the average number of data chunks that an adversarial node can download with the maximum reputation score k is at most $s \cdot \gamma \cdot \bar{x}$, where $\bar{x}$ is the expected number of credits issued by an honest node.*

Proof Sketch The maximum credit diversity of k colluding adversaries is $k \cdot s$ without any upload contribution, hence the maximum reputation score is $k \cdot s$.

Next, we prove the bound on the maximum downloads an adversarial node can perform with maximum reputation. Let X' be the random variable of the number self-issued credits of Sybil identities and let Z' be the random variable of the number of self-issued credits minted by the Sybil identity of a randomly chosen credit among adversarial nodes. We know that $Pr(Z' = x) = \frac{Pr(X'=x)x}{E(X')}$. Since adversarial nodes aim to fairly divide the credits issued by Sybils among themselves, the issuance distribution for each adversary's credit pool can be approximated by the overall distribution Z'.

The filtering process ensures that the resulting credit pool of an adversary passes the set of p_i tests:

$$\begin{aligned} p_i &< Pr(b_i \leq Z' < b_{i+1}) \\ &< \frac{Pr(b_i \leq X' < b_{i+1}) \cdot b_{i+1}}{E(X')} \end{aligned} \qquad (3)$$

Substituting $p_i = \frac{Pr(b_i \leq X < b_{i+1}) \cdot b_i}{E(X)}$ into Inequality 3 and re-arranging sides, we obtain:

$$E(X') \leq \frac{b_{i+1}}{b_i} E(X) \frac{Pr(b_i \leq X' < b_{i+1})}{Pr(b_i \leq X < b_{i+1})}$$

$$= \quad \gamma \cdot E(X) \cdot \frac{Pr(b_i \leq X' < b_{i+1})}{Pr(b_i \leq X < b_{i+1})} \quad (4)$$

In Inequality 4, γ is determined by the number of chosen bins (m) such that $\gamma = \frac{b_{i+1}}{b_i}$ for all i. Moreover, Inequality 4 holds true for all i. As the last step of simplification, we use the property that for any given positive numbers a, b_1, c_1, b_2, c_2, if $a \leq \frac{b_1}{c_1}$ *and* $a \leq \frac{b_2}{c_2}$, then $a \leq \frac{b_1+b_2}{c_1+c_2}$. Applying this observation to Inequality 4 for all i, we obtain:

$$E(X') \leq \gamma \cdot E(X) \quad (5)$$

Because the total number of credits issued by $k \cdot s$ Sybil identities is $k \cdot s \cdot E(X')$, each adversarial node gets at most $s \cdot E(X')$ credits which is less than $s \cdot \gamma \cdot E(X)$. By definition, $E(X)$ is the expected number of credits for the truncated distribution after excluding $\delta \cdot n$ top issuers. Since we assume the number of colluding adversaries is less $\delta \cdot n$, the number of honest nodes is greater than $(1 - \delta)n$. Hence, $\bar{x} \geq E(X)$ where $\bar{x}$ is the expected number of credits issued by an honest node that issues non-zero credit. Thus, we derive $E(X') \leq s \cdot \gamma \cdot \bar{x}$.

It is interesting to note that, for a given range $[b_0, b_m]$ of a truncated distribution of X, the system parameter of the number of bins (m) uniquely determines γ. Specifically, $\gamma = \sqrt[m]{\frac{b_m}{b_0}}$. Therefore, γ decreases as we increase the number of bin m, improving the bound on sustained collusion attacks. However, when m is too large, we also risk filtering out too many credits from a honest node's credit pool unnecessarily.