Peer-to-Peer Networking and Applications (2019) 12:951–968
https://doi.org/10.1007/s12083-018-0701-x

Reputation based approach for improved fairness and robustness in P2P protocols

Francis N. Nwebonyi[1] · Rolando Martins[1] · Manuel E. Correia[1]

Received: 8 November 2017 / Accepted: 19 November 2018 / Published online: 6 December 2018

Abstract
Peer-to-Peer (P2P) overlay networks have gained popularity due to their robustness, cost advantage, network efficiency and openness. Unfortunately, the same properties that foster their success, also make them prone to several attacks. To mitigate these attacks, several scalable security mechanisms which are based on the concepts of trust and reputation have been proposed. These proposed methods tend to ignore some core practical requirements that are essential to make them more useful in the real world. Some of such requirements include efficient bootstrapping of each newcomer's reputation, and mitigating seeder(s) exploitation. Additionally, although interaction among participating peers is usually the bases for reputation, the importance given to the frequency of interaction between the peers is often minimized or ignored. This can result in situations where barely known peers end-up having similar trust scores to the well-known and consistently cooperative nodes. After a careful review of the literature, this work proposes a novel and scalable reputation based security mechanism that addresses the aforementioned problems. The new method offers more efficient reputation bootstrapping, mitigation of bandwidth attack and better management of interaction rate, which further leads to improved fairness. To evaluate its performance, the new reputation model has been implemented as an extension of the BitTorrent protocol. Its robustness was tested by exposing it to popular malicious behaviors in a series of extensive PeerSim simulations. Results show that the proposed method is very robust and can efficiently mitigate popular attacks on P2P overlay networks.

Keywords Trust · P2P · Edge clouds · Reputation · And security

1 Introduction

Computing is currently drifting towards the network edge due to increased interest in decentralized solutions that offer enhanced privacy preserving algorithms, better data locality and alternative communication infrastructures. As an example, there are growing opportunities to engage the resources and properties of mobile devices in creating platforms to harvest resources locally, through the use of hyper local clouds, also called edge clouds [1]. These edge clouds offer a way to reduce dependency on centralized infrastructure (including standard cloud computing infrastructures), especially in highly dense environments such as stadiums, concerts and museums where communication infrastructures are normally under strain.

Generally speaking, P2P distributed networks are able to incorporate remote users by allowing them to connect easily and in an open manner to each other, thereby making communication cheaper, convenient and readily available. They can be very dynamic: self-organizing, self-discovering and self-adapting. This opposes the traditional centralized systems which can be expensive and sometimes lack availability. But several security issues remain unsolved [2], and different forms of aggressive attacks still target online P2P systems [3, 4].

The literature emphasizes trust and reputation based methods as the way forward. This is demonstrated by many trust models which have already been proposed [5–11], targeting different P2P platforms. Successful online commercial services such as Ebay use some of these methods to help users to discern who to transact with, based on their reputation. Kazaa also used a reputation management system, where users rate the participation level of each other, by giving more grade to those that share

✉ Francis N. Nwebonyi
fnwebonyi@dcc.fc.up.pt

Rolando Martins
rmartins@dcc.fc.up.pt

Manuel E. Correia
mcc@dcc.fc.up.pt

1 Faculty of Sciences, University of Porto, & CRACS/INESC-TEC, Porto, Portugal

genuine contents frequently [8]. Similarly, BitTorrent adopts a 'give and take' method known as tit-for-tat. Although this is not exactly a reputation based system, it also rewards users that contribute more to the network.

In this work, we have reviewed some popular trust and reputation models in the literature that are aimed at addressing security challenges in distributed networks and P2P in particular. By identifying and building on their strengths, we derived a method that can address their perceived weaknesses. Based on the review which is discussed in section 4, some of the noticed weaknesses (which also highlights our areas of key contributions) are summarized as follows:

I. Most earlier methods focus mainly on how to mitigate attacks from malicious nodes after they have joined the network, but pay less attention to stopping or limiting malicious peers from joining initially. Those who attempt to overcome this limitation often require pre-established trust relationships over other channels, or some central entity such as server or super nodes which can constitute prime target for attacks. The new method overcomes these barriers and provides a distributed and fairer means of bootstrapping reputation scores for newcomers.

II. Current methods tend to channel huge effort towards protecting client nodes from malicious server nodes (or seeders), but little attention is often given to the protection of server nodes from malicious clients, especially in distributed P2P platforms. This gives rise to a form of bandwidth attack which has escaped the attention of most trust and reputation models in the literature. This attack has been shown to have adverse potential effect on P2P networks such as BitTorrent [12, 13], it is further discussed in section 3. To the best of our knowledge, our work is the first to tackle bandwidth attack (on seeders) in a distributed P2P reputation model, based on actual leecher-to-leecher behavior of peers. The new method empowers both client and server nodes to directly identify attackers and act fairly.

III. It is a common practice to use interaction experience between "trustor" and "trustee" to determine reputation and trust. In some cases, number of good interactions minus the bad ones normally form the bases for direct reputation and trust scores. Other times, it is based on the ratio of good to total number of interactions. But the cumulative interaction rate barely reflects in the end result. This sometimes leads to situations where more familiar node(s) may have little or no noticeable advantage over a less familiar one. In the new method, this silent interaction rate concept (or 'familiarity') is accounted for, as a way of encouraging consistency among nodes. When nodes have similar reputation scores, the new method would give priority to the agent with higher interaction rate, to reward consistency. Each peer is also monitored to see how much they have invested in the network relative to their capabilities and with respect to how much they have gained from it. This concept played a significant role towards mitigating bandwidth attacks. It is further discussed in section 5.2.

In order to assess our approach, we tested it on a classical BitTorrent protocol, via detailed simulations. BitTorrent is used mainly because of its popularity and well-known behavior, but our approach is designed to be adaptable to other similar P2P protocols. Our implementation has been done using PeerSim, an event based simulator [14] which has an implemented version of BitTorrent. The remaining part of this work is organized as follows; an introduction to trust concepts is given in section 2. Section 3, provides an overview of BitTorrent, and describes some well-known attacks on P2P networks, while section 4 discusses some related works. The proposed method is presented in section 5, succeeded by detailed description of experiments and discussion of results in section 6. Conclusion and future work then appears in section 7.

2 Trust concept

Trust has been identified as a crucial concept in digital security, without which it would be difficult to reason about the security of any system in a convincing manner [15]. While it is difficult to narrow trust down to a single universal definition, we adopt the view of Gambetta et al. [16] who defined it as a level of subjective probability on which a party bases his assessment that another party will act in a certain manner, before monitoring the action (or even if he is not able to monitor it), and given that the assessment affects his own action.

To consider an entity trustworthy means to have high probability that such entity will act favorably or at least in a manner that is not harmful, within a given context. Trust is context dependent, more like saying that you can trust a lawyer's legal advice but not his advice on how to fix your car. It is usually based on some sort of relationship; direct, indirect or both.

Reputation is usually a basis for determining trust, which implies that if an entity has maintained a good reputation during past interactions, it can be accorded a level of trust that would warrant subsequent interaction, within a context. Reputation can be seen as a view about an entity within a community, which is generally known and assessed by the members of that community [17].

Broadly speaking, trust can be direct, indirect or hybrid. Direct trust is achieved by observing a neighbor directly, without involving a third party. On the other hand, the concept of trust transitivity is usually harnessed to establish indirect trust, which involves gathering recommendations from neighbors about an entity. The indirect trust of a trustee is dependent on the observation of other entities concerning its behavior during their past interactions. These observations are usually communicated to the trustor in the form of recommendations. A trustor is a node who decides whether or not to trust another peer, the trustee.

There is also a hybrid of these two forms of trust. It involves the combination of direct and indirect trust to achieve even a stronger paradigm. This becomes particularly important when the trustor does not have enough information based on direct experience, to make decision concerning the trustee.

Security has been broadly classified into two categories; hard security and soft security [18]. Hard security is associated with traditional security approaches such as traditional access control and authentication. It provides partial security, because only corporate resources are protected from unauthorized users, but it does not always protect legitimate users from malicious service providers. This means that a central entity (usually the service provider) is able to assess peers that are requesting service from him, but the service requester nodes are not empowered to assess such service provider in a distributed manner. Soft security tackles this weakness, and trust is often promoted as the basis of soft security [19].

2.1 Reputation computation

For nodes to communicate on trust platforms, it is necessary for them to be able to measure the level of trust they can place on each other. This influences the disposition of the nodes to engage in transactions, or otherwise.

It can be challenging to measure digital trust with absolute precision, especially in distributed environments. The fact that many peers posses limited resources and capacities, adds to this challenge. Moreover, trust is a social concept, and some human instincts and reasoning which aid people in making trust decisions, are not easy to capture in algorithmic form. However, the literature contains series of approaches which have been proposed for the purpose of capturing, measuring and computing trust for digital environments [17]. Most of these proposals adopt one (or more) of the following methods to calculate reputations, upon which trust is based.

2.1.1 Summation

The simplest form of trust computation is usually to sum the positive and negative scores separately, then subtracting the negative from the positive, in order to determine how reliable an agent has been. This approach has been adopted to calculate direct reputation in [5], and it is also used by ebay reputation system [20].

2.1.2 Average

Another method which is close to simple summation involves taking the average of all scores of the peer being assessed, such as the case of Amazon [21]. Some other models such as TE-AODV [22] also use this method, assigning default score to newcomers. Sometimes, this method is slightly extended, by taking the weighted average, instead of the normal average. The extended version allows for taking the reputation of the recommending node (or similar factors) into account, as done in [8].

2.1.3 Bayesian Method

Interaction outputs are usually rated as good or bad, that is, in a binary form. The number of positive and negative scores are used to compute reputation, using the beta probability density function (PDF). The PDF tuple α and β represent the number of good and bad interactions respectively. The probability expectation value of beta distribution is represented as follows [23]:

$$E(p) = \alpha/(\alpha+\beta). \tag{1}$$

Compared to other methods, this method tend to have detailed theoretical background which can amount to higher confidence on its outcome. During bootstrapping, α and β are assigned the value of 1 each, which amounts to a default score of 0.5. Despite being popular, this type of (default) score initialization has been criticized for having the likelihood of being either unfair to the new node or the ones already in the network. It also has a tendency of not adequately addressing white washing [24, 25].

2.1.4 Fuzzy Techniques

This is one of the methods adopted for computing and aggregating reputation [8]. It involves reasoning about trust in-terms of fuzzy values. It appears to give more room for incorporating human-like reasoning such as agreement compliance, transaction time/age, etc., in reputation computation and aggregation. Given many recommendations for example, fuzzy technique can be applied to probe for the relevance of such recommendations by reasoning about compliance (or otherwise) to transaction agreement. That is, applying compliance as an objective measure to verify the subjective user ratings [26]. Different techniques apply various means to capture trust, but fuzzy techniques appear popular in this kind of reasoning.

2.1.5 Flow Techniques

Here, reputation has to do with the level of importance that a peer attracts within a community. As an example, in PageRank [27], the number of links pointing to an entity (such as website) compared to the number of links that leaves the entity, is used to determine the importance or reputation of that entity. If we take I(P) to be the set of other entities that are pointing to page P, otherwise called the in-links, and $|P|$ to be the number of other pages that P is pointing out to, otherwise called the outlink. Then PageRank of P denoted by r(P), is as follows:

$$r(P) = \sum_{Q \in I(P)} \frac{r(Q)}{|Q|}, \quad (2)$$

where $|Q|$ is the out-links from Q. r(P) is mainly influenced by the pages that are pointing to it, which can be regarded as recommendations. However, an increase in the number of references made by Q, reduces the influence of Q on P. As the inlinks of Q grow, the rank or reputation of P also grows.

2.1.6 Belief Models

This approach is based on probability theory, but the sum of probabilities for all possible outcomes do not necessarily have to be equal to 1. Trust is expressed as a belief that a system will resist malicious attacks, or that a person will cooperate and not defect. Unlike some other models in which belief in an agent's cooperation is considered true or false (i.e. discrete), belief models consider a case where there is no clear information on either belief or disbelief, and thus an uncertainty factor is introduced. This is done in a way that instead of using belief or disbelief in order to determine trust, uncertainty also counts. Opinions are usually expressed in the form [28]: $\omega_{Y^X} = (b, d, u, a)$, where ω_{Y^X} is X's opinion about statement Y, while b, d, and u stand for belief, disbelief, and uncertainty receptively. a∈[0, 1] is referred to as relative atomicity, which is used to determine how uncertainty affects the expected opinion.

2.1.7 Discrete Trust

Trust is a social concept and thus humans as social entities are often better at determining the trust of other entities, based on experience and other intangible factors. Computationally, scholars try to capture as much of the human related attributes as they can, so as to make algorithmic trust possible and close to the human perception of trust. For example, some models such as [17] have emulated the human style of trust perception, to discreetly rank trust as very trustworthy, trustworthy, untrustworthy, and very untrustworthy. Lookup tables are used to then determine the actual trust of an entity.

3 BitTorrent

BitTorrent is the most popular among P2P protocols, accounting for about 53% of the entire P2P traffic, and more than 30% of overall Internet traffic [29]. It is a good case study because it is well known, widely deployed and it harbors major properties and behaviors of P2P systems, with some added peculiar features.

A share process in BitTorrent is initiated by first creating a meta-data describing the file that is to be shared, including SHA1 hash information about the pieces, among others. The information in the torrent file guides peers to the tracker and subsequently to other leechers and seeders in the swarm [30]. Leechers are peers that do not have all the file pieces, while seeders are peers with all the pieces. A tit-for-tat method is used to ensure that leechers who receive files also give to others. Seeders are assets to the network because they selflessly share files, without the need to download. Because of this, the tit- for-tat does not directly apply to them and thus they can be exploited in some ways, as discussed in the following subsection.

In DHT (Distributed Hash Table) based BitTorrent, the tracker does not aid in the discovery of files or peerset. Nodes do this all by themselves. With distributed tracker systems such as Mainline DHT (MLDHT), peers use content IDs (also called infohash) to find the location of desired contents. Once a node knows the infohash of the file it wishes to fetch, it applies some controls to arrive at its location. Some of the controls include; 'PING' for ascertaining node's availability, 'FIND NODE' used to get the k closest neighbors, 'GET PEERS' for getting the initial peerset and 'ANNOUNCE PEER' used by nodes to announce that they are part of the swarm [31]. In our method, the tracker also does not play any special role; it is focused on individual peers and thus can function without any central agent.

3.1 P2P attacks

Decentralized networks suffer from diverse forms of attack. Apart from free-riding, which is an act of selfishness, in which peers download from others without uploading to them in return, some other popular attacks in BitTorrent and P2P include the following:

1. *Lying Piece Attack:* The goal of this attack is to destabilize the rarest first policy of Bittorrent, in which peers while downloading, give priority to blocks that are less available in the swarm, so that they will not be lost completely. Attackers advertise false pieces, thereby misleading other peers on the pieces that are actually rare [4]. More generally, the intention of attackers here might also be to cause other forms of confusion, such as diverting the attention of victims from genuine contents to non-existing ones, in order to frustrate their download efforts.
2. *Chatty Peer Attack:* Attackers establish many TCP connections with the victims. They announce possession of many file pieces, but when the victims request for some blocks of such pieces, they never upload any. Instead, the attackers resend handshake messages, thereby sticking as

neighbors to the victims, who spend considerable amount of time waiting in vain for the attacker's response [32]. This can be an extension of the previous attack.

3. *Fake-Block Attack:* Attackers also advertise possession of many or all blocks. When victims request for such blocks, they send fake blocks in response. After downloading all blocks of a piece (from attackers and genuine peers), the victim checks if they are genuine. This check fails because of the fake blocks, warranting that the entire piece be downloaded afresh, meaning a huge waste of bandwidth and time [3].
4. *Bandwidth Attack:* This is usually an attack targeted at the seeders with the goal of occupying their upload bandwidth, so that there is little or none left for legitimate peers. Specifically, BitTorrent seeders do not download, so they keep uploading to the fastest downloading peer(s). Attackers therefore exploit this by simply connecting to seeders and downloading only from them at fastest rate possible, so that they are constantly unchoked, while the reputable peers are continually choked [12, 13, 33]. A version of this attack can also apply to other P2P networks, since there are always nodes that serve as resource donors (seeders) in the network. Attackers can target those server nodes in various ways, thereby obstructing their services.
5. *Sybil Attack:* Fake identities, otherwise called sybils can be created very cheaply. To beat reputation systems, attackers create links between its sybils, and through that means unleash different kinds of attack, e.g. bad-mouth attack. They can also use such links for raising their own reputation and thus gaining unmerited advantage [34, 35].
6. *Index Poisoning Attack:* In P2P file sharing platforms such as BitTorrent, clients keep indexes of files which link each file identifier to their hosts. Index poisoning attackers exploit this setting by publishing fake file indexes, with the goal of stopping their victims from accessing genuine neighbors. The attacker basically starts a file sharing task but with fake information such as port numbers, IP addresses, infohash, etc. Since there are no strong verification means, owing to the openness of the system, this attack is easily executed. The innocent peers are left to pay the price by using up their time and resources trying to access contents that are not existing; leading to denial of service [36].
7. *Combination Attack:* This is a form of content pollution attack, which combines fake-block and index poisoning attacks, to make a stronger impact. Attackers in this case extend index poisoning by including their own IDs in the fake infohash to be advertised. The idea is that if the victims insist on establishing connections despite failures due to fake information, they would eventually connect to the attacker(s) who make situation more difficult for them, by feeding them with fake blocks. So, they suffer the effects of both fake block attack and index poisoning attack. Combination attack has been illustrated to impose more harm than fake block or index poisoning attack individually [37].
8. *Peer Exchange (PEX) Attack*: This is very similar to distributed denial of service (DDoS) and index poisoning attacks. It is sometimes captured separately to portray the fact that they are launched on PEX; taking advantage of its features such as large list of peers (up to 3000 in some cases), and very frequent PEX massages. Although this is not the ideal design of PEX, studies have shown that almost all PEX implementations do not follow the original rules and there is no effective way of enforcing it [38]. Making it easy for malicious peers to fabricate huge PEX peer lists with false IP addresses, so victims waste resource and time trying in vain to connect. Similarly, the attacker can also send PEX massages containing the address and port number of the victim to as many others as it can. This is aimed at weighing the victims down with too many connection requests, resulting to DDoS.
9. *Collusion Attack:* Collusion attackers collude to favor themselves at the expense of legitimate peers. Attackers can join forces under collusion attack to make any of the mentioned attacks more devastating [39]. In P2P trust and reputation models, collusion attackers give themselves favorable scores, and downgrade the scores of others. This can mislead genuine peers into erroneously believing that the malicious nodes are the reputable ones, and vice versa. With this trick, attackers can take over the network and successfully eliminate the genuine nodes from network services.

4 Related work

Attempts have been made to mitigate attacks and unfairness in P2P, such as [3, 5, 40, 41]; each with their strengths and weaknesses. In this section, we review some of these reports, beginning with those that are specifically focused on BitTorrent, before exploring P2P in general. We intend to capture popular opinions in the field, and how they have addressed the focus of the proposed method.

A recent work [5] targets free-riding and similar attacks in BitTorrent. Most decisions are made by the tracker, such as decisions about the peers to service at any given time. This makes the method too centralized. The authors of [3] proposed a less centralized method in which 10% of the nodes act as super-nodes, and are charged with the responsibility of decision making. However, there is a possibility of some malicious nodes actually becoming super nodes, and then ruining the network. Moreover, this method also relies on central agents.

Wong et al. [7] mainly focused on content pollution. For any pollution suspected, the network is divided into sub units, in order to identify and carve out the pollutant. This technique can cause huge overhead and possible isolation of network parts. In the same vein, Santos et al. [42] proposed a reputation voting system based on subjective logic, where peers are required to vote for files as either polluted or whole. However, vote collection can take time, since the nodes might need to download big chunk of the file before casting their votes.

Sybil attack was the focus of [43] in Kademlia based BitTorrent. Nodes assess their potential service providers based on trust scores. In particular, trust scores are taken into account when ordering the k-bucket peers. New comers are kick started with some positive risk score to enable them join the network.

As noted earlier, these methods reflect the trend in most P2P reputation and trust models, which involves placing emphases mainly on client nodes, in order to shield them from malicious server nodes, but with minimal emphases on protecting the server nodes (seeders) from greedy leechers. This creates room for bandwidth attack, which has received little or no attention in current trust and reputation models. Some methods which are not reputation or trust based have attempted to address the challenge, but suffer crucial limitations related to poor or nonexistent means of verifying claims/votes [33], and outrageous overhead due to frequent encryption and decryption operations [44].

Another trend that is easily noticeable is that most methods bootstrap trust by assigning initial trust (or risk) score which does not reflect the newcomer's behavior, and which may be unfair to either the existing nodes or the newcomer. Some exceptions are subsequently discussed.

Besides BitTorrent, trust models for other P2P protocols, also portray similar trends. For instance, [8, 10] both initialize new comers with scores that allow them to join the network without initial assessment, thereby making white-washing very cheap. The proposals in [45, 46] require bootstrapping servers for new nodes to join, which can be a bottleneck in distributed settings.

Other methods such as [47] require some form of 'pre-established' trust relationship for newcomers, which may not always be available. An alternative solution which is based on Opennet model allows publicly known 'seednodes' to assist in introducing new nodes. However, this might involve revealing vital information to malicious nodes too, which they can use to attack the network. Additionally, if for any reason a 'seednode' becomes malicious, its effect would be high because it has high privilege. A Sybil resistant method presented in [48] is also based on pre-established trust relationship, which can be unavailable in some cases. Such relationships are expected to have been acquired offline, and are accumulated to form a bootstrap graph, which is a core part of their modified DHT routing mechanism.

In general, proposals that have targeted trust or reputation bootstrapping problem can be categorized into five [49]. The first and most popular method involves assigning default values to newcomers, examples of models belonging to this category were earlier discussed. The second is the dynamic initialization method [50], in which the current level of security within the network (or system) determines the scores that would be assigned to the new nodes. If maliciousness is currently high, then the value of initial trust score will be low, otherwise it would be high.

Thirdly, there is the recommendation and endorsement based method. Under this category, if a newcomer have similar interest and capabilities [24] or patterns [25], with an existing service (or node) which is already known as credible, then its initial score can be derived from such node or service. The fourth involves the use of game theory [51] for predicting initial trust, usually in web service platforms. While the fifth involves the use of central/super entities [52], or prior registration [49], or pre-existing trust relationship which was mentioned earlier [47, 48].

Another method involves the use of challenge/puzzle [53, 54] to restrict the number of fake identities that an attacker can introduce into the network. This is less trust-based and more related to resource testing. Since existing nodes need to verify the puzzle in order to introduce a newcomer, they may be placed at a disadvantage if the would-be attacker has more resources. Our method is different because the existing nodes do not have such disadvantage. The new method is also distributed, with no need for prior registration, pre-existing trust relationship or super node. It also addressed bandwidth attack which has been omitted in earlier systems.

The discussed related works are summarized in Tables 1 and 2. In Table 2, 'Y' stands for 'yes', '-' represents 'no', 'NA' means 'Not Applicable', while 'P' is for 'partial'. Our judgment for Sybil attack on the table agrees with an earlier work in [34], and more recently [43] which noted that no effective Sybil attack measure has been derived for distributed networks. We used the term 'partial' to mean that the reviewed method can minimize the attack, but not stop it completely. While 'yes' indicates those we think have addressed them more intensely. For want of space, we have not included methods that focus only on Bootstrapping in the tables, they have however been discussed previously.

5 Proposed method

The proposed approach is termed FBit (Fairer BitTorrent), it is aimed at achieving better fairness and robustness against maliciousness in P2P systems such as BitTorrent, with focus on bandwidth attack, fake block attack, Sybil attack and collusion attack. FBit ensures that every peer is treated with priority that reflects its level of cooperation, thereby providing better motivation for cooperation, better resource management, network

Table 1 Summary of related work

Ref.	Strength	Weakness
[9]	Copyright protection.	Pre-trusted and central entities.
[8]	Effective aggregation.	Resource intensive due to broadcasts.
[11]	Partial anonymity.	Not accountable for peers that drop requests.
[10]	Scores reflect global view.	Pre-trusted peers has to be trusted always.
[5]	Hard on free-riders.	Poor score sieve.
[3]	Scorer's reputation accessed.	Rogue nodes might make it to super level.
[33]	Independent decision	Fake votes easy.
[40]	Vote falsification is hard.	Resource intensive.
[42]	Early assessment.	Not DHT compatible.
[41]	Automatic scoring.	Only implicit, not established.
[7]	Checks fake blocks.	Tracker could be overworked.
[6]	Detects fake piece.	False positive/negative.
[43]	Distributed.	Only manages the effect of sybils.
[47]	Trust bootstrapping.	When pre-relationship is unavailable, the alternative can be weak.
[48]	Safe routing.	New comers require pre-trusted peers.
FBit	Distributed, efficient bootstrapping, bandwidth attack mitigation.	Churn effect not analyzed.

safety, fairness and efficiency. Although BitTorrent has been used for illustration here, the new method can be applied to similar P2P systems. The following subsections contain detailed explanation of the various components of the proposed method.

5.1 Reputation bootstrapping

At the point of entry, every node is expected to generate a key pair (*Kp*, *Kb*) with which they sign their transactions. New nodes start by discovering other nodes that are already in the swarm through the tracker (or DHT), in a usual BitTorrent pattern. Afterwards they send "Bitfield" (or similar control) massages requesting to be added as neighbors. Through the message, they also send their self-generated public keys and self-signed certificates to show that the request originated from them. When a trustor receives such request, it associates the accompanying public key with the trustee, to distinguish it from other nodes. For the newcomer's request to be granted, it has to demonstrate some preliminary trustworthy acts. Node's ID can be derived from its public key.

Table 2 More on summary of related work

Ref.	Bootstrapping Assessment	Fully Distributed	Content Pollution	Selfish Acts	Sybil Attack	Bandwidth Attack
[9]	–	–	Y	–	P	–
[8]	–	Y	Y	Y	P	–
[11]	–	Y	Y	Y	P	–
[10]	–	Y	Y	Y	P	–
[5]	–	–	Y	Y	P	–
[3]	–	–	Y	Y	P	–
[33]	NA	Y	–	–	–	Y
[40]	–	Y	Y	Y	P	–
[42]	–	–	Y	P	P	–
[41]	–	–	Y	P	P	–
[7]	–	–	Y	Y	–	–
[6]	–	Y	Y	Y	P	–
[43]	–	Y	Y	Y	Y	–
[47]	Y	Y	Y	Y	Y	–
[48]	Y	Y	–	Y	Y	–
FBit	Y	Y	Y	Y	Y	Y

As an illustration, assuming that *nodeA*, *nodeC* and *nodeE* were already in the network, when *nodeB* requests to join through *nodeA*. After *nodeB's* request, when *nodeA* receives the next "unchoke" message from any of its neighbors (e.g. *nodeC*), it responds by sending a request for a block, according to BitTorrent procedure. However, in addition, it requests that the block should be routed through *nodeB* (which is the new node), instead of having it sent directly. When *nodeB* receives such block from *nodeC*, it forwards it to *nodeA*, appending its signature, based on the key it generated initially.

Given that the transaction is honest, *nodeB* gains some reputation point, and the reputation of *nodeC* is updated too. This ensures that the new node does not gain reputation advantage over the existing ones. *NodeA* can repeat this step for some other blocks, giving *nodeB* more opportunities to serve and build reputation. In case of DHT, proximity would be considered in the routing process, such that nearby nodes would be more active in helping a newcomer to gain reputation.

While *nodeA* is waiting, it can proceed with other requests so that it is not trapped if *nodeB* does not deliver. Other nodes in the network such as *nodeC* and *nodeE* can also route some of their requests through *nodeB* in order to speed-up the bootstrapping process. The new node is not allowed to request any service until it is admitted. Each transaction, both from the new and old nodes are signed. We count on the reputation of the introducing nodes to expect that they will act fairly to the newcomers. However, they can loose reputation for inappropriate introduction, and can also gain for appropriate ones.

When *nodeB* joins through multiple nodes (e.g. *nodeA*, *nodeC*, *nodeE*), *nodeA* determines if *nodeB* is due to be added as a neighbor, by collecting recommended Bootstrap factors (*Bf*) from the other nodes. *NodeA* further weighs each recommendation based on its trust on the nodes that sent them, as illustrated in Eq. (3). *BfCB* is the *Bf* reported by *nodeC* concerning *nodeB*, while G*SAC* is the reputation of *nodeC* from the perspective of *nodeA*. Same pattern repeats for every neighbor that would submit Bf recommendation. Notice that *nodeA* still adds its own bootstrap factor, but with an optimal self-reputation of 1:

$$\mathrm{Bf}_{\mathrm{A_B}} = \left(\sum\nolimits_C \mathrm{Bf}_{\mathrm{CB}}.\mathrm{GS}_{\mathrm{AC}}\right) + \mathrm{Bf}_{\mathrm{AB}}. \tag{3}$$

No further normalization is relevant at this stage because the newcomer is not expected to service too many nodes before it joins, and not all neighbors are therefore expected to respond to a request for *Bf* recommendation. Notice also that the interest is on the cumulative value, not the average. We pass the accumulated *Bf* through a unit step function, *Θ(Bf)* which returns either 0 or 1 depending on the value it receives, according to Eq. (4), where n_{min} is the minimum *Bf* required from a newcomer:

$$\Theta(\mathrm{Bf}) = \begin{cases} 0 \text{ if } < \mathrm{n_{min}} \\ 1 \text{ otherwise} \end{cases}. \tag{4}$$

The self-signed certificates mentioned previously, does not stop generation of multiple fake identities, but it helps to ensure that malicious nodes do not impersonate existing trustworthy ones. Recall that new nodes usually do some work in order to gain entry into the network, which is to service some existing node(s) up to a minimum limit. This would mean spending some resources and a bit of time by the newcomers. Since resources are not always limitless, the number of fake identities that a Sybil attacker can introduce into the network is limited. The attackers are further frustrated in the network through the reputation and 'interaction rate' checks which are subsequently discussed.

It is important to minimize the number of fake identities at entry stage because the lesser they are in number, the lesser their impact. The difference between the work done by newcomers in FBit and that which they do by solving puzzles, is that genuine FBit nodes which are already in the network do not have to spend extra resources to verify the work done by newcomers. They simply perform normal checks which they usually do for all massages, even those from known trustworthy nodes. So, there is less burden (due to bootstrapping) on the trustors and the network in general. Whitewashing also appears less appealing and thus nodes are discouraged from becoming malicious after joining the network.

Similarly, the new approach does not suffer the kind of fundamental problem [55] experienced in puzzle based methods, which has to do with disparity in resources or computation abilities between legitimate users and would-be attackers. With the new method, even large disparity in computational abilities between legitimate nodes and the new comers (or potential attackers) is not of visible significance. Moreover, the proposed method favors productivity, because newcomers actually render valuable help, and in return gain initial reputation. As explained in subsection 5.4, this method can be adapted to similar other file sharing P2P platforms. It is distributed, and does not rely on any seed node or super peer.

5.2 Familiarity

Many P2P file sharing systems (e.g BitTorrent) operate in a "give and take" manner. This is not strange since the network is usually self-sustaining. The idea behind familiarity (or interaction rate) is to enable nodes to ascertain how much each neighbor has contributed to them particularly, and to the network in general. We assume that more familiar nodes interact more often, and if genuine, contribute more to each other. This idea is similar to the tit-for-tat method of BitTorrent where nodes prioritize neighbors that have given more to them.

However, unlike BitTorrent where nodes act mainly based on local tit-for-tat views, the new method allows nodes to also ascertain the global impact of each node on every other node in the network (or neighborhood) at a given time, in-terms of resource contribution, on a scale of 0 to 1. This ensures that a more liberal and reputable node, gains some advantage, not just for its reputation but also for its liberality in terms of rendering services to other nodes. We shall focus on familiarity in this subsection, and then capture reputation in the next.

Equation (5) captures the activity rate of peers in a swarm (which is a kind of P2P network) at intervals ($t = 20$ s). If node i downloads from node j, or uploads to it, i records an interaction (download or upload), and vice versa. We also take note of unanswered requests, because they could be signs of maliciousness, such as lying piece attack. The time when interactions occur are also recorded. Given that node i has x successful interactions (or transactions) with node j within time t, it calculates the interaction rate of node j; $IR_{ij}(t)$ using Eq. (5):

$$\mathrm{IR}_{\mathrm{ij}}(\mathrm{t}) = \mathrm{X}_{\mathrm{ij}}/\mathrm{n}_{\max} \tag{5}$$

where n_{max} is derived by dividing the node's bandwidth by the size of each block and multiplying the result by time (t). Literally, n_{max} is the optimal rate at which each peer is expected to function in the swarm within t interval, and it is recalculated every t seconds. An alternative way to determine n_{max} in an environment where knowing the bandwidth is difficult, would be to collect votes from participating nodes based on their interactions during their first few seconds in the network. Nodes pay more attention to neighbors who service them in return. If extended, it means that nodes give priority to neighbors who give back to the network, in order to ensure sustainability. Successful transaction means non malicious uploads and downloads, which are expected to be considerably mutual.

As familiarity grows, IR tends to 1. n_{max} is chosen in a way that keeps IR below 1 within the time (t) range. However, in cases of extreme familiarity, depending on the choice of n_{max}, IR can be greater than 1. When this happens (which can be rare in practice), the excess is not considered, so IR simply equals 1.

As already noted, most earlier approaches use parameters such as the difference between number of a peer's *'downloads from'* and *'uploads to'* another peer as a way of determining cooperation. For instance, if node i has uploaded z chunks to node j and downloaded y chunks from it, then y minus z will be the bases for determining the fairness or otherwise of j, from the perspective of i. In some cases, ratio is used instead of the difference, mostly in models that are based on probability expectation value [23]. y and z can also represent the number of favorable and unfavorable interactions respectively. Unanswered requests are often counted among the unfavorable transactions.

The problem with such method is that the cumulative interaction rate does not reflect adequately on the end results. As an example, consider a case where $y = 500$, and $z = 499$, the difference will be 1, just the same as when $y = 2$ and $z = 1$, ignoring the interaction frequencies, which can be a vital factor by itself. If we take the ratio of good interactions over total, then the case with fewer interaction will be at clear advantage, which can be somewhat unjust. Wang P. et al. [56], identified similar problem, but their approach soldered the concept rigidly into trust computation, such that it might be a problem when emphasis is preferred either on just trust or interaction rate.

In a nutshell, we use this concept of familiarity to capture the activity rate of a peer, not just locally, but generally in the network, on the scale of 0 to 1. Trust score alone can hardly reveal this relationship. We have applied it more specifically in subsection 5.5 to tackle bandwidth attack.

5.3 Reputation score computation

Applying the concept of probability expectation value [23], and based on recorded interaction experience, non-malicious downloads are associated with α, while the bad ones plus uploads (as well as no replies) are associated with β. Expected behavior based on the previous reputation (DT_{ij}) of node j, according to the direct experience of node i with respect to time (t), is given in Eq. (6):

$$\mathrm{DT}_{\mathrm{ij}_{(t)}} = \frac{\alpha_{\mathrm{ij}_{(t)}} + \Theta\left(\mathrm{Bf}_{\mathrm{i}_{\mathrm{j}}}\right)}{\alpha_{\mathrm{ij}_{(t)}} + \beta_{\mathrm{ij}_{(t)}} + \Theta\left(\mathrm{Bf}_{\mathrm{i}_{\mathrm{j}}}\right)}. \tag{6}$$

A download is considered non malicious if its piece is hashed without error. Otherwise, if there is a mismatch in the hash code, it is considered malicious. Bf serves as a normalizing factor when computing expected reputation, since every node in the neighborhood of i is expected to have been bootstrapped, which implies that it has $Bf > 0$. The number of 'uploads' counts for β because we are interested in both credibility and contribution rate. For example, the reputation of a free-rider can drop as a result of its act of not giving. While those that barely give, would be caught up in the familiarity check.

As the number of interactions grow, the behavior of nodes may change, making it important for older interactions to have less weight or be completely forgotten. After more than 1 interactions (i.e. if $\exists\ DTij(t-1)$), decay or aging factor is introduced. The aging factor used here is related to that of [8] and it basically considers the similarity between previous records and the current outcome, the wider the difference, the less consideration such history is given. If we denote the aging factor by $\rho^{(t)}$, then we update our local experience with respect to the just concluded transaction at time (*t*), as given in Eq. (7):

$$\mathrm{DT}_{\mathrm{ij}_{(t)}} = \rho^{(t)}\mathrm{DT}_{\mathrm{ij}_{(t-1)}} + \left(1-\rho^{(t)}\right)\mathrm{DT}_{\mathrm{ij}_{(t)}}, \quad (7)$$

this equation also applies to IR.

To get a general view of the reputation and familiarity of any neighbor, nodes usually ask others for recommendations concerning that neighbor. Replies to such recommendation requests usually come in pairs; DT and IR. Considering our testbed (BitTorrent), nodes can also update the tracker with recommendation information when they make contacts, although such tracker update is not a requirement for the proposed method. When a node asks for recommendation from its neighbors, it ranks each recommendation according to the reputation of the recommending node, using the Ordered Weighted Average (OWA) [57]. The choice of OWA is due to its weight tuning advantage, it allows us to easily associate each indirect score with the reputation of the node that sent them. If node *i* asks its *n* sequence of neighbors (*j)* about another peer *l*, *i* discounts collected recommendations (DT_{jl}), and interaction rates (IR_{jl}) as indicated in Eqs. (8) and (9):

$$\mathrm{IT}_{\mathrm{il}_{(t)}} = \frac{\sum_{j=1}^{n}\mathrm{DT}_{\mathrm{ij}_{(t)}}\mathrm{DT}_{\mathrm{jl}_{(t)}}}{\sum_{j=1}^{n}\mathrm{DT}_{\mathrm{ij}_{(t)}}}, \quad (8)$$

$$\mathrm{CIR}_{\mathrm{il}_{(t)}} = \frac{\sum_{j=1}^{n}\mathrm{IR}_{\mathrm{ij}_{(t)}}\mathrm{IR}_{\mathrm{jl}_{(t)}}}{\sum_{j=1}^{n}\mathrm{IR}_{\mathrm{ij}_{(t)}}}, \quad (9)$$

where $\mathrm{IT}_{\mathrm{il}_{(t)}}$ and $\mathrm{CIR}_{\mathrm{il}_{(t)}}$ are the reputation and *IR* of *l* respectively, according to *i*, based on the information it got from other peers. And assuming that $\mathrm{DT}_{\mathrm{jl}_{(t)}}$ scores are arranged in descending order. Similarity check, elaborated in section 6, is performed on the collected recommendations before computation. This is to sieve out recommendations that may have been submitted by collusion attackers.

The trustor's local reputation about the trustee $\left(\mathrm{DT}_{\mathrm{il}_{(t)}}\right)$ can be added using Eqs. (8) and (9), with peak weight of 1. Alternatively, recommended scores (IT, CIR) and direct scores (DT, IR) can be distinct, and then merged with Eqs. (10) and (11), adjusting the weight ($0 \le \sigma_d \le 1$) to suit peculiar needs. We adopted the later (using Eqs. (10) and (11) for the results shown in this work, with a weight of 0.6 assigned to σ_d. In our experience, this value appears optimal because it gives reputation an upper hand, without undermining familiarity factor:

$$\mathrm{TR} = \sigma_\mathrm{d}.\mathrm{DT} + (1-\sigma_\mathrm{d}).\mathrm{IT}, \quad (10)$$

$$\mathrm{TIR} = \sigma_\mathrm{d}.\mathrm{IR} + (1-\sigma_\mathrm{d}).\mathrm{CIR}, \quad (11)$$

where TR stands for Total Reputation, and TIR is the Total Interaction Rate. They can further be merged with Eq. (12). We observed on the course of this work that it makes lots of difference when IR is considered, compared to when it is not. ($0 \le \sigma_t \le 1$) is a weighting factor, in Eq. (12):

$$\mathrm{GS} = \sigma_\mathrm{t}.\mathrm{TR} + (1-\sigma_\mathrm{t}).\mathrm{TIR}, \quad (12)$$

σ_t has same value as σ_d mentioned previously.

5.4 Modified BitTorrent unchoke algorithm for leechers

For clarity, we have divided the algorithm into two parts. The first (algorithm 1) briefly discussed in this subsection, captures the steps that FBit leechers take when unchoking other leechers. While the second part (algorithm 2) which is presented in the next subsection, depicts the steps that seeders take when unchoking leechers. Algorithm 1 applies earlier discussed concepts of bootstrapping, familiarity and reputation to mitigate Sybil attacks, fake-block attacks, free-ridding, and similar others. While algorithm 2 focuses specifically on stopping bandwidth attacks. The two algorithms work smoothly together.

Some acronyms used in the algorithms which were not earlier defined are:

- *MaxUnchoke*; used to indicate the maximum number of neighbors a server node can service simultaneously at a given time.
- *NumberUnchoked*; used to show the number of neighbors that is currently being served by a given node.
- *InterestedPeerList*; keeps a list of neighbors that are interested in a given block of the file.
- *TrustedPeerList*; used to keep a list of nodes (among those interested in the current block) whose reputation score is above *'threshold'*. IR does not influence this score; *'GS'* only counts for peers that have not been found malicious at this preliminary stage.
- *threshold*; minimum score (0.5) a peer must have before it can be added to the list (*'InterestedPeerList'*) of those that will be considered for a given transaction.

Algorithm 1 Leecher Unchoke Method

```
 1: $MaxUnchoke = c$
 2: $NumberUnchoked = 0$
 3: $InterestedPeersList \leftarrow P$
 4: $TrustedPeerList \leftarrow \{\}$
 5: while $NumberUnchokded \leq MaxUnchoke$ do
 6:     for all peers(P) in InterestedPeerList do
 7:         Calculate IR and DT scores()
 8:         if $IR$ Is High then
 9:             $TR = DT$
10:             $TIR = IR$
11:         else
12:             Get recommendations and check similarity
13:             Apply equations (8) to (11)
14:             Calculate GS() according to (12)
15:         end if
16:         if $TR \geq threshold$ then
17:             $TrustedPeerList \leftarrow p$
18:         end if
19:         Sort TrustedPeerList by GS()
20:         Unchoke Top Scores First()
21:         $NumberUnchoked{+}{+}$
22:     end for
23:     Remove P from InterestedPeersList
24: end while
```

In a nutshell, here is the function of each line in the first algorithm. Lines 1 and 2 initiates *MaxUnchoke* and *NumberUnchoked* variables respectively, while lines 3 to 4 start the lists of *InterestedPeerList* and *TrustedPeerList*. According to line 5, if a node is currently able to service any additional neighbor, it selects a trustee from the *InterestedPeerList* (line 6) and calculates the *IR* and *DT* of the trustee (line 7). If the trustee is very familiar to the trustor (line 8) then scores are based on direct experience only (lines 9 and 10), otherwise recommendations are collected (lines 12) and used for calculating scores (lines 13). When recommendations are collected, a similarity check (explained in the last paragraph of section 6) is also done. *GS* is afterward calculated in line 14. However, if a trustee has a TR that is below threshold, it does not qualify to be considered for any service (lines 16 to 18). Those who are qualified to receive services are serviced according to their *GS* scores beginning with the highest score (lines 19 and 20). When a transaction process is initialized, the trustee is removed from the *InterestedPeerList* (line 23).

When nodes are able to validate each other and complete transactions, they mutually update reputations and interaction rates, reflecting their experiences. In algorithm 1, leechers use such updates in determining requests to respond to, and the priority that each requester deserves. Equations (6) and (7) are applied to update direct reputation and familiarity (rate) information. If a node has communicated regularly with consistent reputation for a considerable amount of time (even recently), then it makes sense to apply local information in making decision for a subsequent transaction, in order to save resources that would otherwise be used to gather and compute recommendations.

Trustors contact neighbors for recommendations when they do not have enough information to assess the trustee, or when the direct familiarity between the two nodes is not high enough. Line 13 of algorithm 1 triggers Eqs. (8) to (11) which are responsible for computing indirect scores and combining them with direct experience of the trustor. The general score (GS) is further calculated using Eq. (12) which basically applies desired weights to merge reputation and familiarity scores. Based on the GS, all trustees that are not considered malicious are collected in a list and priority is given to each according to its contribution and reputation in the network.

Although peers may be given different names in different P2P platforms, the concept of seeder (service giver) and leecher (service receiver) is quite common. In most cases, especially in distributed P2P, network peers serve as both service providers and consumers. The algorithms presented in this work has not been built to rigidly fit into BitTorrent topology alone. Focus has been more on the nodes, so that any similar topology would require minimal tuning to adapt it.

5.4.1 FBit in DHT based systems

DHT (including MLDHT) makes it possible for the network to function without a tracker. Every node acts as a mini tracker; they collectively perform the task of discovering other peers in the network. In MLDHT, nodes randomly choose 160-bit unique ID which also indicates their distance (in a way). At first, nodes need to query the k-bucket to get nodes that are closest to the infohash. The client node further tries to connect to those nodes in order to initiate queries. Given that the connected node is aware of the peers that are associated with the infohash, it returns a list of such peers and file download continues similar to the way it happens in the traditional protocol [58].

At this point, just before the download processes, the reputation bootstrapping can happen for newcomers. Key exchange messages can be embedded in any of the controls mentioned earlier (eg. 'FIND_NODE'), while newcomers may be served after successfully responding to some requests such as request for peers that are associated with the infohash (which the newcomer should have at this stage). In the current implementation of the proposed system, the tracker does not perform any special task, every node is regarded as equal. This is a way of making it non-centralized and enabling its adaptability to other platforms.

FBit is focused more on content downloads (retrieval/storage) and less on routing, but it can be featured in both processes. Instead of considering only the distance in choosing the k-bucket nodes, *GS* score in FBit can be engaged in the process. A weighted average of the *GS* and "distance factor" can be adopted, similar to the method used in [43]. With the advantages being reputation bootstrapping, computation of indirect reputation and mitigation of bandwidth attack.

Similarly, in Gnutella, servants perform the task of both servers and clients [59]. To join, Gnutella newcomers connect to any known host, and through that means get in-touch with other nodes. Such known host(s) can serve as the initial trustor(s) in the bootstrapping stage of the proposed system. Similarly, the "broadcast" and "back-propagation" of information in Gnutella can serve as means of relating the reputation (and other information) of peers to neighbors within the network. The new method can also be adapted easily to fit into other offspring of P2P such as mobile edge-clouds since every node is capable of making independent (and informed) decision.

5.5 Modified BitTorrent unchoke algorithm for seeders

As mentioned earlier, malicious peers take advantage of the opening discussed under bandwidth attack to abuse seeders and download mainly from them, thereby blocking their unchoke slots (i.e. making them unable to upload to legitimate peers). Such behavior also enable the malicious nodes to avoid downloading from other leechers who could measure their contributions based on experience. To check this, FBit in algorithm 2 warrants that a leecher (*l*) who requests download from a seeder (*s*), has to submit its top *IR* information alongside its request. *s* uses such information to determine how much *l* has uploaded and downloaded from its fellow leechers.

The seeder then gives priority to peers that are more selfless in their service to other leechers, in order to motivate them and encourage others not to be selfish. It also mitigates a kind of denial of service attack that would occur if the unchoke slots of seeders are congested by attackers. This algorithm further holds each peer responsible for not only its reputation, but also the impact it is able to make in the network. Recall that seeders are basically service givers who have no need for receiving. P2P networks need them, and they need to be encouraged and optimally protected.

Algorithm 2 Seeder Unchoke Method

```
1: MaxUnchoke ← {predefined}
2: InterestedPeersList ← {}
3: NumberUnchoked ← 0
4: while NumberUnchokded ≤ MaxUnchoke do
5:     CollectVotesIncludingRecentlyCached()
6:     DoIntermitentScoreValidation()
7:     CountVoteForEachP()
8:     UnchokeHighestVotedP()
9:     if PeersHaveSameVote() then
10:        UnchokePWithHigherIRScore()
11:    end if
12:    NumberUnchoked++
13:    RemovePFromInterestedPeersList()
14:    if VotedP Is Not In InterestedPeersList then
15:        CacheVote()
16:    end if
17: end while
```

As an illustration, nodes *l*, *i*, and *j* in Fig. 1 represent individual peers in a swarm who request services from a seeder node. The message accompanying their requests indicate other peers (leechers) that they have downloaded the most from, and the ones they have uploaded the most to. A malicious node who chooses to download only from seeders in order to dodge assessment by fellow leechers, will have little or no "upload" information to give and thus will not be qualified to make requests. In this illustration, node *i* will be served first because it has the highest vote, before node *l* and then node *j*.

Seeders confirm scores by validating submitted scores for consistency. For instance, it could be seen that nodes *i* and *l* have both submitted scores about each other at same time interval; it is therefore expected that the download rate (*DR*) score which *i* submitted concerning *l* at a given time should match the upload rate (*UP*) score that node *l* submitted concerning *i* at about the same time. Line 6 of algorithm 2 executes this check, every *t* seconds.

If two or more peers have the same number of votes among the submitted top leechers, then the highest IR score will be given priority, according to lines 9 to 11 of the algorithm. If their *IR* scores are still same, the first request will be served first. By giving priority to nodes that show more cooperation with fellow leechers, they are rewarded for being cooperative and others are encouraged to do likewise. Low score nodes which are genuine will have no problem downloading from other leechers in order to step up their scores. At the initial stage however, services by seeders are on first-come, first-served basis.

Seeders also cache some of the information temporarily, so that it could be used in the near future if need be. For example, if any other node besides *i*, *j* and *l* were voted, its score will be saved for when such node would request a service, within some time limit (lines 14 to 16 of algorithm 2).

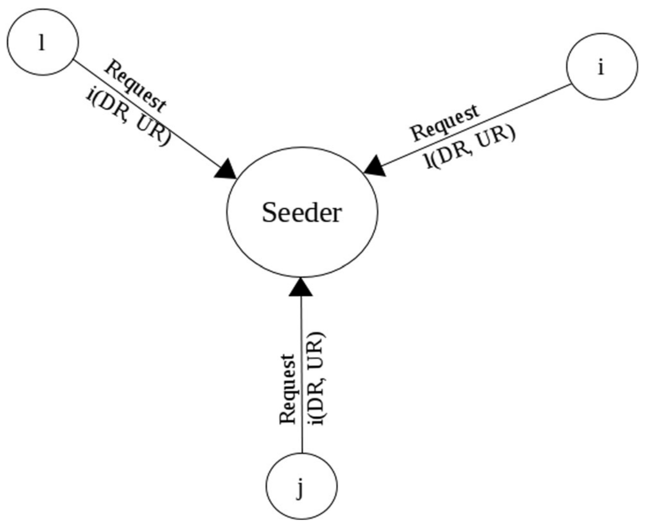


Fig. 1 When leechers send requests to a seeder

Here is a summary of the functions of the remaining lines in algorithm 2. Lines 1 to 3 initiates the variables the same way it is done in the first algorithm, and line 4 also has similar function. Line 5 enables a server node to collect votes from its client nodes indicating the *IR* information of other nodes they have interacted with. Line 7 counts valid votes for each client node in the *InterestedPeerList* and then services are rendered based on active cooperation in the network (line 8). Once a transaction process is initiated, line 12 updates the number of nodes that are currently being served and line 13 removes the node from *InterestedPeerList* to indicate that it is being served. After each transaction round, the *MaxUnchoke* is decreased.

6 Experiments and results

The PeerSim simulator was used to implement a testbed for the proposed method. It is a java based P2P simulator which already includes a BitTorrent protocol implementation [14, 60]. This allowed us to focus only on the adaptation of the protocol to our approach, while retaining the original version for comparison.

The simulation runs on two virtual machines each with 16 CPUs at 2500 MHz and 64GB RAM. Each machine runs OpenJDK Runtime Environment 1.8, and Python 3.4.3 which is used for scripting. PeerSim has a configuration file that allows parameters to be adjusted as desired. Some of such parameters include;

- *network.size;* used to specify the network size. This was set to 100.
- *network.node.direct_weight*; this is a new addition, used to specify the weight of direct scores (that is, *IR* and *DT*). It was set to 60%.
- *protocol.urt* was set to "UniformRandomTransport", which is the transport protocol used by the simulator.
- *protocol.simulation.file_size*; this is used to specify the size of file to be shared in the network. The process is completely successful if every peer downloads a complete file before the process ends. It was set to 20mb in the simulation.
- *protocol.simulation.duplicated_requests*; this was set to 1, meaning that a node can only send one request for a particular block at a time. It could be set to any other value if desired.
- init.net.*seeder_number*; this was set to 1, so that each simulation begins with one seeder. There are other variables that are initialized at the start of the simulation such as the number of attackers. When a type of attack is not present, its value is set to 0 (e.g. init.net.*nCollusionAttacker 0*). There are also some controls that allow us to observe and get feedback from the simulation (e.g. control.observer.step simulation.logtime). During "logtime", we update the files that have been created to store the information needed to measure the performance of the network.
- *protocol.simulation.max_swarm_size* was set to 200 to show that the network can only grow to a maximum of 200 nodes (there is no specific reason for choosing 200, except that we needed a number that is more than the network size which is 100). This can change to accommodate any network size of choice.
- *random.seed*; every simulation needs to have a seed value. Using the scripts, this and other necessary variables are automatically generated at the beginning of each simulation. In the case of the random seed, the script generates it randomly, while other values (such as percentages of attacker) are picked accordingly from the series of provided values.

A 95% confidence level was maintained for the recorded simulation results, this is in order to statistically validate their consistency. Simulations were ran with network size of 100 nodes, and varied number of malicious nodes, to determine the effect in each case. Different attacks were simulated including sibyl, fake-block and collusion attacks, with the goal of determining how efficiently such attacks are mitigated by FBit compared to other methods. The original BitTorrent method [60] and another Trust Management System (TMS) [3] were adopted for comparison. For TMS, we implemented it using the information provided in [3]. TMS was chosen for comparison because it addressed similar attacks (such as fake-block attack), and was also tested on BitTorrent platform.

The simulation began with a single seeder in each case, and runs until a 95 percentage confidence level is attained, after which an average result is gathered. Each simulation can run for a maximum of 30 times. The file size is 20mb for all experiments. This work did not address the implications related to churn, nodes stay in the network throughout the simulation. The presented results were obtained with network size 100 and varied percentages (0, 10, 20, 30, 40, and 50, 60, 70, 80) of attackers.

Figure 2 presents a result of how the network responded to Sybil attackers, who also distribute fake-blocks when they are able to gain access. The result compares FBit to the original tit-for-tat based BitTorrent and TMS. As shown, non-malicious nodes suffer less attack with FBit compared to other methods; they are able to download at a significantly faster rate. The proposed bootstrapping method helps to minimize the introduction of fake identities (IDs) into the network. Introducing a fake ID is made costly and thus a Sybil attacker can only introduce a limited number, based on its capacity. FBit further fishes out such limited number of fake identities through reputation and familiarity checks.

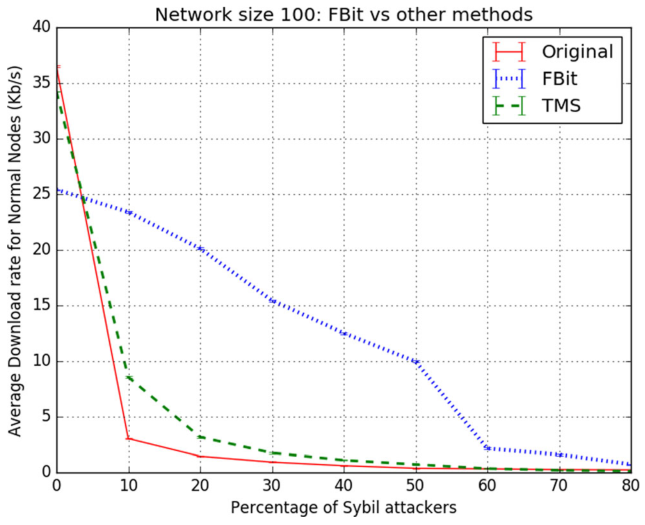


Fig. 2 FBit vs other methods; showing the difference in download rate for non-malicious nodes. FBit allows them to download at higher rate amidst Sybil attackers

When the number of attackers rose beyond 50% of the network size, the graph (Fig. 2) indicates that their effect on the FBit network became stronger, and mitigating them led to a sharp decrease in the download rate of legitimate nodes. This is expected because, with such high percentage, there will be more failed and repeated transactions. Transactions fail when nodes suspect that their neighbor is likely an attacker, and decline its offers. When this repeats, more time will be needed to find the right nodes, and thus the overall transaction time will increase.

Reputable nodes which are already in the network do not share in the cost of introducing or validating newcomers, except for slight ‘transaction time trade-off’ that may arise as a result of routing some chunks or queries through the newcomer. Such tradeoff is however compensated by the gain in reputation that the trustor stands to get after successfully introducing a new node. The delay caused by forwarding some packets (or queries) through the newcomers appears mild. If legitimate existing nodes had to spend computational resources for challenge verification, as would be the case in puzzle based methods, the impact would be more significant (at-least from logical point of view).

TMS could not cope with Sybil attackers because of the earlier mentioned flaw in the bootstrapping method. If fake IDs can be introduced almost at no cost, then an attacker can easily introduce them in numerous number and hijack the network at early stage. In tit-for-tat, new comers are admitted without checks via optimistic unchoke, giving room for attackers who sap the available resources and frustrates the network with fake blocks.

FBit was also exposed to collusion attacks, where fake-block attackers cooperate to favor themselves and downgrade the reputation of others that are not in their clique. Figure 3 captures the performance of FBit in comparison with tit-for-tat based BitTorrent and TMS. A careful look at the result reveals that FBit can cope with collusion attacks. Although collusion attackers appear to have higher impact compared to a non-collusion scenario such as Fig. 2, FBit nodes are still able to complete download at considerably fair rate. The noticed decline in download rate (with higher percentage of attackers) is expected, because finding reliable nodes become more difficult and more time consuming. The amount of genuine pieces available in the network also declines as the number of genuine nodes decrease. The TMS method felt more collusion impact, while the original method was completely overwhelmed.

To guard against collusion attack, TMS uses recommendations from top 10 nodes to compute the scores of the other peers. From the result shown, this does not appear effective because malicious nodes also stand a chance of joining the top nodes, especially with exaggerated scores from other colluding nodes. Tit-for-tat also does not effectively combat collusion attack because nodes are overwhelmed by malicious traffic from the attackers. For example they can be trapped in a cycle of downloading, verifying and discarding fake blocks; making it impossible for them to successfully complete transactions. Moreover, in TMS and original methods, attackers are able to steal resources from seeders through bandwidth attack.

Some models have applied neighbor similarity to detect attackers in P2P. For example, in [61], if recommendations collected from various neighbors concerning a peer, lack similarity, then such peer is regarded as an attacker. Their approach was designed for e-commerce, and it requires coordination from some form of central agents such as group leaders, which can be seen as a limitation when considering more distributed platforms. However, we tapped from their idea of similarity, and applied it (with modifications) to tackle collusion attack.

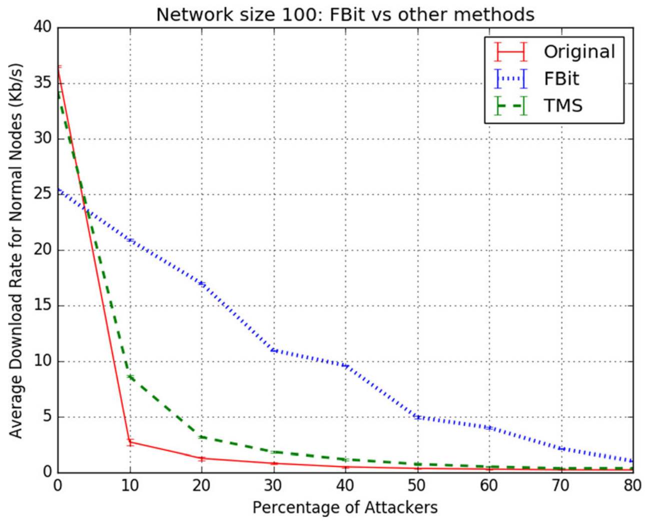
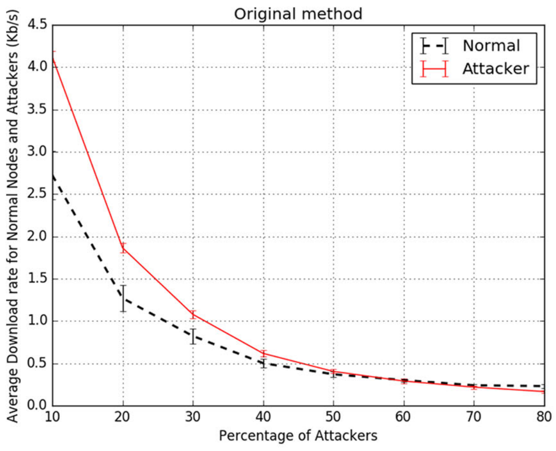
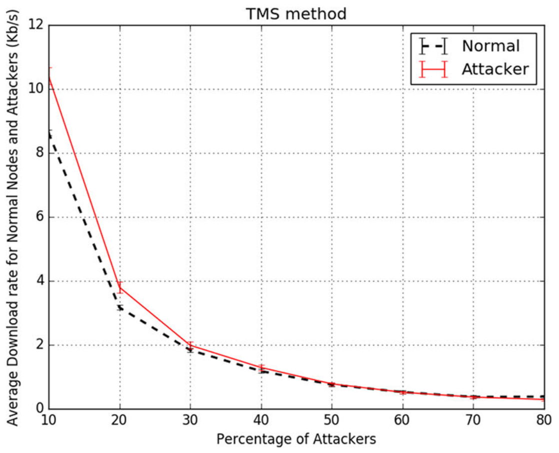


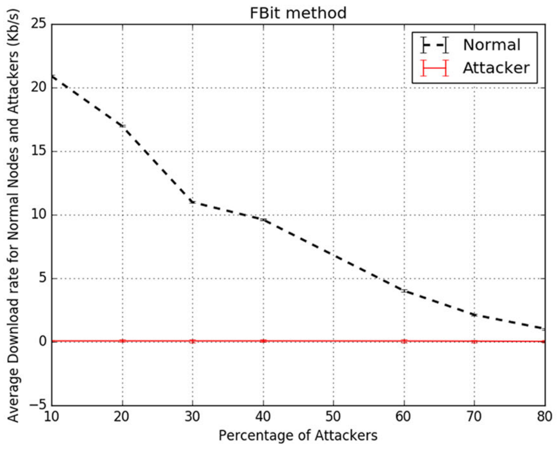
Fig. 3 Collusion attack



(a) Original



(b)TMS



(c) FBit

Fig. 4 The download rate of genuine nodes versus attackers in the original (**a**), TMS (**b**) and FBit (**c**) methods. A Resilient system is expected to frustrate attackers and allow non malicious nodes to download faster. In the original and TMS methods, attackers are able to download, especially from seeders. TMS did better than the original method, but was still tricked by the attackers. FBit successfully stopped the attackers from stealing resources (from seeders), showing resilience against bandwidth attack

When a peer is sending a request for recommendation, it includes a randomly selected trusted neighbor(s) from its neighbor list (which must be different from the neighbor being inquired about, and the one from whom recommendation is required). So it requests recommendation for the actual unfamiliar peer, and at least one familiar fellow that would be used to check similarity.

If the recommendation giver is in a collusion clique with the unfamiliar peer, then it will give it high score, and downgrade the reputable one. A malicious recommendation giver can also decide to simply exaggerate both scores, but it will still be noticed in the similarity check. When recommendations from various neighbors are gathered, priority is given to the ones that are most similar, based on the known peers that have been inquired about. Line 12 of algorithm 1 triggers the similarity check.

To measure how successful an attack is, we captured how the malicious nodes fared compared to non-malicious ones. Ideally, non-malicious nodes are expected to beat the attackers and maintain a clearly higher download rate. Figure 4 illustrates this, with non-malicious nodes downloading clearly at higher rate in FBit, while the collusion attackers were dominating in the other methods. The gap is more in the original method and less in TMS, indicating that TMS is more resilient to collusion attack than the original method.

7 Conclusion

We have devised a bootstrapping approach for BitTorrent and similar P2P protocols. It is distributed and appears efficient with the simulation test cases. In contrast to the popular method of assigning default reputation scores to newcomers, our method provides them with a distributed avenue to work for their initial reputation, before they are able to fully join the network and request services. No central entity or pre-existing relationship is required.

Similarly, the proposed algorithm adequately shields seeders from the effects of bandwidth attacks. This is possible through a modified familiarity approach that makes seeders aware of peers' leecher-to-leecher relationships. Seeders are important asset in P2P and need to be protected adequately, but as we highlighted, they receive minimal attention before now and therefore they face exploitation. The proposed model promises improved network stability, security, fairness and efficiency.

We are continuing research on distributed P2P along several directions. As a follow up to this work we are exploring the concept of Root Cause Assessment (RCA) [62], to see how they can empower nodes to deal with irregular behaviors, such as cases where nodes alternate between good and bad behaviors irregularly. There is also an ongoing plan to fit FBit into an edge cloud platform such as [63, 64].

Acknowledgements This work is partially funded by project "NanoSTIMA: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/NORTE-01-0145-FEDER-000016" financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

References

1. Khan, A.M., Freitag, F. Rodrigues, L.: Current trends and future directions in community edge clouds. In: 4th IEEE International Conference on Cloud Networking (CloudNet), pp. 239–241. IEEE, Niagara Falls (2015)
2. Baqer K, Anderson R (2015) Do you believe in tinker bell? The social externalities of trust. In: Cambridge international workshop on security protocols, pp. 224–236. Springer
3. Sarjaz BS, Abbaspour M (2013) Securing BitTorrent using a new reputation-based trust management system. Peer-to-Peer Networking and Applications 6:86–100
4. Konrath, M. A. Barcellos, M. P. Mansilha, R. B. : Attacking a swarm with a band of liars: evaluating the impact of attacks on bittorrent. In: 7th IEEE international conference on peer-to-peer computing, pp. 37–44. IEEE (2007)
5. Naghizadeh A, Razeghi B, Radmanesh I, Hatamian M, Atani RE, Norudi ZN (2015) Counter attack to free-riders: filling a security hole in BitTorrent protocol. In: 12th IEEE international conference on networking, sensing and control, pp. 128–133. IEEE
6. Dhungel P, Wu D, Ross KW (2009) Measurement and mitigation of BitTorrent leecher attacks. Comput Commun 32:1852–1861
7. Wong KY, Yeung KH, Choi YM (2009) Solutions to swamp poisoning attacks in BitTorrent networks. In: 1st international MultiConference of engineers and computer scientists, pp. 360–363. IMECS
8. Aringhieri R, Damiani E, Vimercati D, De Capitani S, Paraboschi S, Samarati P (2006) Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. J Am Soc Inf Sci Technol 57:528–537
9. Qureshi, A. Rifa-Pous, H. Megıas, D.:Electronic Payment and Encouraged Cooperation in a Secure and Privacy-Preserving P2P Content Distribution System. In: The 7th International Conferences on Advances in Multimedia, pp. 8–14. MMEDIA(2015)
10. Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The eigentrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th international conference on world wide web, 640–651. ACM press
11. Cornelli F, Damiani E, di Vimercati S, Paraboschi S, Samarati P (2002) Choosing rep- utable servents in a P2P network. In: Proceedings of the 11th international conference on world wide web, pp. 376–386. ACM press
12. Dhungel P, Hei X, Wu D, Ross KW (2008) The seed attack: can bittorrent be nipped in the bud?. Technical report, Department of Computer and Information Science. In: Polytechnic institute of NYU
13. Dhungel P, Hei X, Wu D, Ross KW (2011) A measurement study of attacks on bittorrent seeds. In: 2011 IEEE international conference on communications (ICC), pp. 1–5. IEEE
14. Montresor A, Jelasity M (2009) PeerSim: A scalable P2P simulator. In: 9th IEEE international conference on peer-to-peer computing, pp. 99–100. IEEE
15. Nwebonyi FN, Ani UP (2015) DanielBYOD network: enhancing security through trust– aided access control mechanisms. International Journal of Cyber-Security and Digital Forensics 4:272–290
16. Gambetta D (2000) Can we trust trust?. Trust: making and breaking cooperative relations. In: Gambetta, Diego (ed.) trust: making and breaking cooperative relations, electronic edition, Department of Sociology, University of Oxford, pp. 213–237. University of Oxford
17. Jøsang A, Ismail R, Boyd C (2007) A survey of trust and reputation systems for online service provision. Decis Support Syst 43:618–644
18. England P, Shi Q, Askwith B, Bouhafs F (2012) A survey of trust management in mobile ad-hoc networks. In: Proceedings of the 13th annual post graduate symposium on the convergence of telecommunications, networking, and broadcasting. PGNET
19. Lilien L, Al-Alawneh A, Ben Othmane L (2010) The pervasive trust foundation for security in next generation networks. In: Proceedings of the 2010 workshop on new security paradigms, pp. 129–142. ACM
20. Resnick P, Zeckhauser R, Swanson J, Lockwood K (2006) The value of reputation on eBay: a controlled experiment.: experimental economics, pp 79–101. Springer
21. Gregg DG (2009) Outline reputation scores: how well are they understood?: journal of computer information systems, pp 90–97. Taylor & Francis
22. Venkanna U, Agarwal JK, Velusamy RL (2015) A Cooperative Routing for MANET Based on Distributed Trust and Energy Management. In: A cooperative routing for MANET based on distributed trust and energy management.: wireless personal communications, pp. 961–979. Springer
23. Josang A, Ismail R (2002) The beta reputation system. In: Proceedings of the 15th bled electronic commerce conference, pp. 2502–2511. Bled
24. Skopik F, Schall D, Dustdar S (2009) Start trusting strangers? Bootstrapping and prediction of trust. In: International conference on web information systems engineering, pp. 275–289. Springer Berlin Heidelberg
25. Yahyaoui H, Zhioua S (2011) Bootstrapping trust of web services through behavior observation. In: International conference on web engineering, pp. 652–659 springer Berlin Heidelberg
26. Sherchan W, Loke SW, Krishnaswamy S (2006) A fuzzy model for reasoning about reputa- tion in web services. In: Proceedings of the 2006 ACM symposium on applied computing, pp 1886–1892. ACM

27. Benincasa, C., Calden, A., Hanlon, E., Kindzerske, M., Law, K., Lam, E., Rhoades, J., Roy, I., Satz, M., Valentine, E., Whitaker, N.: Page Rank Algorithm. : Department of Mathematics and Statics, University of Massachusetts, Amherst, Research (2006)
28. Josang A (1999) Trust-based decision making for electronic transactions. In: Proceedings of the 4th Nordic workshop on secure computer systems, pp. 496–502. NORDSEC
29. Pouwelse J, Garbacki P, Epema D, Sips H (2005) The bittorrent p2p file-sharing system: measurements and analysis. In: International workshop on peer-to-peer systems, pp. 205–216. Springer
30. Fattaholmanan A, Rabiee HR, Large-Scale Active A (2016) Measurement study on the effectiveness of piece-attack on BitTorrent networks. IEEE Trans Dependable Secure Comput 13: 509–518
31. Wang L, Kangasharju J (2013) Measuring large-scale distributed systems: case of bittorrent mainline dht: IEEE thirteenth international conference on peer-to-peer computing (P2P), pp. 1–10. IEEE
32. Dhungel, P., Wu, D., Schonhorst, B., Ross, K. W.: A measurement study of attacks on BitTorrent leechers. In: 7th international conference on peer-to-peer systems, pp. 7–15. ACM (2008)
33. Adamsky F, Khayam SA, Jäger R, Rajarajan M (2014) Stealing bandwidth from BitTorrent seeders. Computers & Security 46: 126–140
34. Douceur JR (2002) The Sybil attack. In: International workshop on peer-to-peer systems, pp. 251–260. Springer Berlin Heidelberg
35. Alice C, Eric F (2005) Sybilproof reputation mechanisms. In: Proceedings of the 2005 ACM SIGCOMM workshop on economics of peer-to-peer systems. ACM Press, USA, pp 128–132
36. Kong J, Cai W, Wang L (2010) The evaluation of index poisoning in bittorrent. In: Second international conference on communication software and networks, pp. 382–386. IEEE
37. Kong J, Cai W, Wang L, Zhao Q (2010) A study of pollution on BitTorrent. In: The 2nd international conference on computer and automation engineering (ICCAE), pp. 118–122. IEEE
38. Su, M., Zhang, H., Fang, B., Du, X.: DDoS vulnerability of BitTorrent peer exchange extension: analysis and Defense In: 2012 IEEE International Conference on Communi- cations (ICC), pp. 1048–1052. IEEE(2012)
39. Saini, N. K., Chaturvedi, A., Yadav, R.: Identifying Collusion Attacks in P2P Trust and Reputation systems.: Int J Comput Appl(IJCA) (2014)
40. Ragab-Hassen, H., Jones, O., Galanis, N.: Rabit: a reputation architecture for BitTorrent. In: 2012 IEEE global communications conference (GLOBECOM), pp. 850–855. IEEE (2012)
41. Ormándi R, Hegedus I, Csernai K, Jelasity M (2010) Towards inferring ratings from user behavior in BitTorrent communities. In: 19th IEEE international workshop on enabling technologies: infrastructures for collaborative enterprises (WETICE), pp. 217–222. IEEE
42. Santos FR, da CC, Weverton L, Gaspary LP, Barcellos MP (2011) Funnel: choking polluters in bittorrent file sharing communities. In: 8th IEEE transactions on network and service management, pp. 310–321. IEEE, vol 8, pp 310–321
43. Riccardo P (2016) A trust and reputation method to mitigate a Sybil attack in Kademlia. Comput Netw 94:205–218
44. Wang J, Wu X, Guo N (2010) Ullrich, C.,Luo, H.: discouraging improper exploitation against seeds in BitTorrent swarms. In: International conference on cyber-enabled distributed computing and knowledge discovery (CyberC), pp. 235–242. IEEE press
45. Singh A, Liu L (2003) TrustMe: anonymous Management of Trust Relationships in Decen- tralized P2P systems. In: Proceedings of the 3rd international conference on peer-to-peer computing (P2P 2003), pp. 142–149. IEEE press
46. Chen K, Liu G, Shen H, Qi F (2015) Sociallink: utilizing social network and transaction links for effective trust management in P2P file sharing systems. In: IEEE international conference on peer-to-peer computing (P2P). IEEE Press, Boston, pp 1–10
47. Clarke I, Sandberg O, Toseland M, Verendel V (2010) Private communication through a network of trusted connections: The dark freenet. https://www.researchgate.net/profile/Vilhelm_Verendel/publication/228552753_Private_Communication_Through_a_Network_of_Trusted_Connections_The_Dark_Freenet/links/02e7e525f9eb66ba13000000/Communication-Through-a-Network-of-Trusted-Connections-The-Dark-Freenet.pdf. Accessed 2 Mar 2017
48. Danezis G, Lesniewski-Laas C, Kaashoek MF, Anderson R (2005) Sybil-resistant DHT routing. In: European symposium on research in computer security, pp. 305–318. Springer
49. Yu Y, Xia C, Li Z (2015) A trust bootstrapping model for defense agents. In: IEEE international conference on communication software and networks (ICCSN), pp. 77–84. IEEE press
50. Tavakolifard M, Knapskog SJ (2011) Trust evaluation initialization using contextual in- formation. In: Proceedings of the international conference on Management of Emergent Digital EcoSystems, pp. 1–8. ACM
51. Jiao H, Liu J, Li J, Liu C (2011) A framework for reputation bootstrapping based on reputation utility and game theories. In: 10th IEEE international conference on trust, security and privacy in computing and communications (TrustCom), pp. 344–351. IEEE press
52. Malik Z, Bouguettaya A (2009) Reputation bootstrapping for trust establishment among web services. IEEE Internet Comput 13:40–47
53. Oram A (2001) Peer-to-peer: harnessing the power of disruptive technologies. O'Reilly me- dia. In: Inc
54. Mónica D, Leitao J, Rodrigues L, Ribeiro C (2009) On the use of radio resource tests in wireless ad hoc networks. Technical report, proc. In: 3rd WRAITS
55. Borisov N (2006) Computational puzzles as Sybil defenses. In: 6th IEEE international con- ference on peer-to-peer computing, pp. 171–176. IEEE press
56. Ping W, Jing Q (2007) A mathematical trust model in e-commerce. In: International conference on multimedia and ubiquitous engineering (. MUE'07), pp. 644–649. IEEE press
57. Yager RR (1988) On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Transactions on systems, Man, and Cybernetics 18:183–190
58. Xinxing Z, Zhihong T, Luchen Z (2016) A measurement study on mainline DHT and magnet link: IEEE international conference on data science in cyberspace (DSC), pp. 11–19. IEEE
59. Ripeanu M, Foster I, Iamnitchi A (2002) Mapping the gnutella network: properties of large- scale peer-to-peer systems and implications for system design. In: arXiv preprint cs/0209028
60. Fabrizio F, Pedrolli M (2008) A BitTorrent module for peersim. University of Trento, Technical report
61. Wang G, Musau F, Guo S, Abdullahi MB (2015) Neighbor similarity trust against sybil attack in P2P e-commerce.: IEEE transactions on parallel and distributed systems, pp. 824–833. IEEE
62. Ferreira A, Huynen J, Lenzini G, Koenig V (2015) In cyber-space no one can hear you S-CREAM: a root cause analysis of technique for socio-technical attacks. In: 11th workshop on security and trust management, pp. 255–264. ESORICS
63. Marinelli E (2009) E.: hyrax: cloud computing on mobile devices using MapReduce. Carnegie-mellon univ Pittsburgh PA school of computer. science
64. Rodrigues J (2017) Marques, E. RB: lopes, L.: Silva, F.: towards a middleware for mobile edge-cloud applications. In: Proceedings of the 2nd workshop on middleware for Edge Clouds & Cloudlets. Pp. 1. ACM

Francis N. Nwebonyi completed his masters degree, MSc Computer security and forensics, from University of Bedfordshire, United Kingdom. His MSc thesis was on trust based approach for security in BYOD networks. He obtained his first degree, BSc computer Science, from Ebonyi State University in Nigeria. Francis worked shortly as an assistant lecturer at Ebonyi State University – Nigeria, after his Msc, before moving to Portugal where he is currently undertaking his PhD, in the department of computer science, University of Porto. He is a member of CRACS (Center for Research in Advanced Computing Systems), under INESC-TEC. He is currently researching on trust based security for distributed networks, such as P2P, and edge cloud.

Rolando Martins studied at Faculty of Science of the University of Porto (FCUP), where he also obtained his M.Sc in Informatics: Networks and Systems. He also worked at EFACEC as a software architect and later as a systems researcher. He obtained his Ph.D in Computer Science from FCUP, as a part of a collaborative effort between FCUP, EFACEC and Carnegie Mellon University, under the supervision of Prof. Fernando Silva, Prof. Luís Lopes and Prof. Priya Narasimhan. His Ph.D. research topic arose from his employment at EFACEC, where he was exposed to the difficulties underlying today's railway systems and light-rail deployments, and came to understand the scientific challenges and the impact, of addressing the issues of simultaneously supporting real-time and fault-tolerance in such systems. He is a former member of the the Intel Science and Technology Center (ISTC), where he was involved in both Cloud Computing and Embedded Computing centers, and Parallel Data Lab (PDL) at Carnegie Mellon University (CMU). At the same time, he was also a computer research scientist at YinZcam, a spinoff from CMU that provided mobile applications for the NBA, NHL, NFL and MLS, where he was involved on cloud computing, content management systems, OAuth and video streaming. He is currently an invited assistant professor at the department of Computer Science at FCUP and researcher at CRACS (Center for Research in Advanced Computing Systems) part of INESC-TEC.

Manuel Eduardo Correia has a Phd in Computer Science from the University of Porto and a MSc in Computer Engineering from Imperial College. He is one of the Co-Founders of HLTSYS and is a Professor at the computer science department of the Faculty of Science at Porto University. He is also the director of the Masters on Informatics security, member of the executive committee of the computer science department and member of the digital administration council of the University. As a security expert, he is also a member of the CT 199 Portuguese normative group, responsible for the translation and adoption of ISO and CEN eHealth related standards in Portugal. He has an extensive experience in research projects, at the national and European level, some of them as PI. He has also acted as consultant for the Portuguese Ministry of Health, in (1) European projects Epsos and SEHGovIA; (2) National security normatives and policies for Health Institutions; (3) Electronic prescription system security and for the Ministry of Justice, where he his responsible for the development of the biometric match-on-card system, currently being deployed into the Portuguese citizen card. (http://orcid.org/0000-0002-2348-8075).