Free-riding, fairness, and firewalls in P2P file-sharing

Please check the document version of this publication:

Free-riding, Fairness, and Firewalls
in P2P File-Sharing

J.J.D. Mol, J.A. Pouwelse, D.H.J. Epema, and H.J. Sips
Department of Computer Science
Delft University of Technology
P.O. Box 5031, 2600 GA Delft, The Netherlands
{j.j.d.mol, j.a.pouwelse, d.h.j.epema, h.j.sips}@tudelft.nl

Abstract

Peer-to-peer file-sharing networks depend on peers uploading data to each other. Some peers, called free-riders, will not upload data unless there is an incentive to do so. Algorithms designed to prevent free-riding typically assume that connectivity is not a problem. However, on the Internet, a large fraction of the peers resides behind a firewall or NAT, making them unable to accept incoming connections. In this paper, we will prove that it is impossible to prevent free-riding when more than half of the peers are firewalled, and we will provide bounds on the sharing ratios (defined as the number of bytes uploaded divided by the number of bytes downloaded) of both firewalled and non-firewalled peers. Firewall puncturing techniques are complex but can be used to connect two firewalled peers; we will provide a bound on their required effectiveness in order to achieve fairness.

We confirm our theory by simulating individual BitTorrent swarms (sets of peers that download the same file), and show that the theoretical bounds can be met in systems with many firewalled peers. We have also collected statistics covering thousands of BitTorrent swarms in several communities, both open and closed; the latter ban peers if their sharing ratios drop below a certain treshhold. We found 45% of the peers to be firewalled in the closed communities, as opposed to 66% in the open communities, which correlates with our theory that to obtain fair sharing ratios for all peers, at most half of them can be behind firewalls.

1 Introduction

Most peers in a P2P file-sharing network will not contribute by uploading files if no countermeasures are taken [1]. Peers which download files without contributing anything in return are called *free-riders*, and one of the ways to avoid it is to give peers an incentive to upload as much data as they download. A common metric of fairness is the *sharing ratio* of a peer, which is defined as the total number of uploaded bytes divided by the total number of downloaded bytes. Some (closed) P2P communities have a sharing ratio enforcement policy, in which peers are banned if their sharing ratio drops below a certain treshhold. Since the user has little control over the amount of data he can upload to other peers while he is downloading a file, he is expected to increase his sharing ratio when necessary by continuing to upload after the download has completed, or by injecting new content.

In order to be able to upload a file to other peers interested in the file, a peer has to be able to establish connections with them. However, in most P2P networks, a large fraction of the peers is behind a firewall or NAT and, as a result, they cannot accept incoming connections. Firewalled peers are therefore unable to connect to each other. We will prove that it is impossible to enforce fair sharing ratios, that is, sharing ratios equal to 1, if more than half of the peers are firewalled. As the percentage of firewalled peers increases, the average sharing ratios of the firewalled peers rapidly drop, regardless of the amount of seeding they do or the amount of content they inject. This implies that free-riding cannot be avoided in such situations, which has a fundamental impact on algorithms that expect peers to be able to contribute as much as they consume.

Some P2P networks use techniques like *firewall puncturing* or *NAT traversal* [4, 7, 8] to enable two firewalled peers to establish a connection. No perfect puncturing technique is known, because firewall and NAT behaviour is not standardised. If the puncturing techniques are effective enough, a fair sharing ratio is possible for all peers. We will give a lower bound on the required effectiveness.

We validate our model using simulations as well as real-world measurements. We simulate individual BitTorrent sessions, and analyse the actual average sharing ratios of the

firewalled and the connectable peers. For real-world measurements, we have collected and analysed data on the behaviour of several BitTorrent communities, both open and closed. In an open community, anyone can join any swarm (the set of peers downloading the same file), without sharing ratio enforcement. In a closed community, only members can download files, and they are banned if their sharing ratio drops below a certain treshhold. On average, the BitTorrent protocol rewards good uploaders with a better download performance. Firewalled peers are limited in the amount of data they can contribute to the system as our theorems will show. We therefore expect the connectivity problems of the firewalled peers to be reflected in their download performance. Furthermore, in a closed community, where fair sharing ratios are enforced for all peers, a majority of firewalled peers cannot be allowed. We will present data on the behaviour of real systems consistent with these observations.

This paper is organised as follows. In Section 2, we will explain the basics behind firewalls and NATs. Then, in Section 3 we will introduce the network model we consider. In Section 4, we will derive bounds on the sharing ratio of both firewalled and connectable peers if no firewall puncturing or NAT traversal techniques are employed. In Section 5, we derive a lower bound on the effectiveness of the puncturing techniques to make a fair sharing ratio possible. In Section 6, we evaluate the sharing ratios of peers in BitTorrent sessions through simulation. In Section 7, we will present our data on the behaviour of real BitTorrent communities. Finally, in Section 8 we discuss related work and in Section 9, we draw our conclusions.

2 Firewalls and Puncturing

Although P2P algorithms are often designed assuming that every peer can accept incoming connections from other peers, in practice this is not always the case. In this section, we discuss the two main causes of this, which are firewalls and NATs, along with firewall puncturing techniques, which allow some firewalls and NATs to nevertheless accept incoming connections anyway.

2.1 Firewalls and NATs

A firewall is a piece of software or hardware which can be configured to block certain incoming connections. Firewalls are used for security purposes — services which are unintentionally exposed to the outside world can be a target for hackers. By using a firewall, the system administrator can decide which protocols, ports, or programs are allowed to accept inbound connections. As a result, some peers in a P2P network which operate behind such a firewall are unable to accept incoming connections.

A Network Address Translation gateway (NAT) is a router which translates network addresses between two address spaces. Such a setup is common if a consumer or corporation is given a single (public) IP address but has several computers he wishes to connect to the Internet. In that case, one computer (the NAT) is assigned the public IP address. All computers, including the gateway, are assigned a private IP address. The Internet traffic is routed through the NAT. For every outbound connection, the NAT keeps track of its origin within the private address space and routes all packets in either direction accordingly. For every inbound connection, the NAT cannot know to which computer the corresponding traffic must be routed.

A NAT is a popular default setup for broadband users, who are often unaware of this connectability problem or do not have the technical knowledge to configure their NAT. Because broadband users form a significant fraction of the users on the Internet, we conjecture that they form the bulk of the firewalls and NATs as well. Some NATs can be configured to route certain traffic to certain computers automatically by the use of UPnP [14], which is a 1999 standard to let desktop computers configure the NAT. However, the adoption of UPnP has been very slow. A 2007 measurement by Xie et al. [22] found only 19% of the peers to have UPnP enabled, even though 89% of the peers were firewalled or behind a NAT.

There are three reasons to assume that not all pairs of hosts will be connected. First, UPnP may never be deployed on all NATs, and will be shipped disabled by default on others. Second, corporations will likely view UPnP to be a security hole, as it allows users to open a port within the private network to the rest of the Internet. Third, NATs have a side-effect of increasing security, because the rest of the Internet cannot access the computers behind the NAT directly. Once NATs can be configured to make such computers connectable, new security threats will arise and firewall usage will increase in response. A common firewall policy is to allow any outbound connections and block all incoming connections except to the services which are explicitly allowed. The firewalls are unlikely to be configured to allow any application to receive incoming connections from anywhere on the Internet, which will invariably lead to a certain fraction of peers with limited connectivity. The fraction of firewalled peers within a P2P system will thus continue to depend on the nature of the community.

We will, for ease of discussion, use the term 'firewalled peers' for both peers behind a firewall and those behind a NAT whose firewall or NAT is not configured to accept incoming connections. Such peers can initiate outbound connections, as well as upload and download to others. They cannot, however, connect to other firewalled peers, unless special techniques are used.

2.2 Firewall Puncturing

Techniques called *firewall puncturing* and *NAT traversal* can be used to establish a connection between two firewalled peers, which typically works as follows. Under the supervision of a (connectable) coordinating peer, both firewalled peers initiate a connection at the same time. When an outgoing connection is initiated, the NAT gateway knows where incoming packets should go if it judges them to be part of the same connection. When using UDP, which is a stateless protocol, a reasonable percentage of the firewalls can be punctured (for example, using STUN [19] or NUTSS [9]). Being able to puncture with stateful protocols like TCP is substantially harder [4, 7, 8], since both of the firewalls need to agree on the state to be established. This involves guessing, as either peer has to predict what state on the other end will be expected. Ford et al. measured an overall success rate of 82% for UDP puncturing and 64% for TCP puncturing [7], although their results varied wildly between different NAT hardware vendors. Note that the success rate will be lower if bidirectional communication is required between firewalled peers, as both peers need to support the implemented puncturing technique.

Although puncturing using UDP has a reasonable success rate, its use complicates P2P system design as it will have to implement its own stateful protocols on top of UDP. Some firewalls or NATs cannot be punctured because they do not allow puncturing for security reasons. Finally, NAT behaviour is not standardised, making the implemented techniques difficult to maintain and their future effectiveness uncertain.

3 Model and Notation

We consider a P2P network consisting of a set $\mathcal{N}$ of peers which will upload and/or download a file (or video stream) of L bytes. As only the amount of data exchanged will be relevant, it will not matter when these peers arrive or depart. The set $\mathcal{N}$ is split up into two disjoint sets, the set $\mathcal{F}$ of firewalled peers, which cannot accept incoming connections, and the set $\mathcal{C}$ of connectable peers, which can accept incoming connections. We assume a peer $p \in \mathcal{F}$ and $q \in \mathcal{C}$ can always connect by having the connection originate from p. Without puncturing techniques, no connections between peers in $\mathcal{F}$ are possible. Furthermore, we define $N \equiv |\mathcal{N}|$, $F \equiv |\mathcal{F}|$ and $C \equiv |\mathcal{C}|$, and we define $f \equiv F/N$ as the fraction of firewalled peers.

We will use two metrics for fairness. First, we define $S_{\mathcal{P}}$ to be the average sharing ratio of a peer in set $\mathcal{P}$. Second, we define the *debt* $\Delta(p)$ of a peer p is the number of bytes downloaded minus the number of bytes uploaded by p, and $\Delta(\mathcal{P}) \equiv \sum_{p \in \mathcal{P}} \Delta(p)$ is the debt of a group of peers $\mathcal{P}$. We will only consider the sharing ratios and the debts of the peers once all peers have completed the download.

The amounts of data contributed by the individual peers is typically skewed: some peers upload more than they download, and other peers download more than they upload. To obtain a fair resource contribution for all peers, *sharing ratio enforcement* can be introduced. The sharing ratio of a peer is the total number of bytes it has uploaded divided by the total number of bytes it has downloaded. A P2P system can be designed to expect users to aim for a sharing ratio of 1, representing a fair situation in which a peer on average has contributed as much as it has consumed. To obtain a fair sharing ratio, a peer can either inject new content or upload the file to others (including seeding, that is, continuing to upload the file after the download is completed).

4 No Firewall Puncturing

This section will provide bounds on the sharing ratio of both the firewalled and connectable peers if the P2P system does not employ firewall puncturing or NAT traversal techniques. The reasoning will be roughly as follows. Firewalled peers can only receive data from connectable peers. If there are more firewalled peers than connectable peers, the majority of firewalled peers can only upload to the minority of connectable peers, and a fair sharing ratio for the firewalled peers will be impossible to obtain. First, we will derive bounds on the sharing ratios of both firewalled and connectable peers. Then, we will discuss some practical implications of the derived results.

4.1 Sharing Ratio Analysis

It is useful to first derive the debt of each peer after all peers have finished downloading the file.

Lemma 1. *For the debt of the firewalled peers* $\Delta(\mathcal{F})$, *we have* $\Delta(\mathcal{F}) \geq ((2f - 1)N - 1)L$.

Proof. We consider two cases: the file has been injected by a firewalled peer, or by a connectable peer. In the first case, since the firewalled peers cannot form connections among each other, they have to obtain the file from the connectable peers. This creates a data flow from $\mathcal{C}$ to $\mathcal{F}$ of size $(F-1)L$ (the injector already has the file). The connectable peers can obtain the file from both sets of peers, creating a data flow from $\mathcal{F}$ to $\mathcal{C}$ of at most $C \cdot L$. Figure 1 illustrates these flows. The difference between these flows is thus bounded by $\Delta(\mathcal{F}) \geq (F - 1)L - C \cdot L = ((2f - 1)N - 1)L$. In case the file was injected by a connectable peer, similar to the first case, we find $\Delta(\mathcal{F}) \geq F \cdot L - (C - 1)L = (f \cdot N - (1 - f)N + 1)L > ((2f - 1)N - 1)L$. ☐

Note that the bound in Lemma 1 holds regardless of whether the injector is connectable or firewalled. The firewalled peers will end up with a positive debt ($\Delta(\mathcal{F}) > 0$) if $f > 0.5 + 0.5/N$. In that case, it is impossible for the
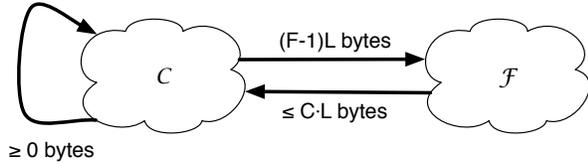
Figure 1. Data flows between connectable and firewalled peers, if the file was injected by a firewalled peer.
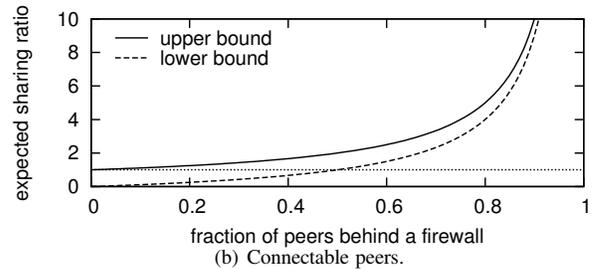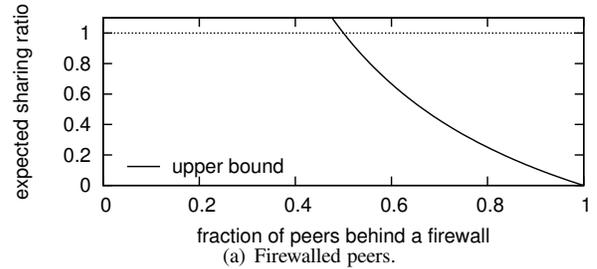

(a) Firewalled peers.

(b) Connectable peers.

Figure 2. Bounds on the average sharing ratio of a peer, against the fraction of firewalled peers. The horizontal dotted lines represent a fair sharing ratio of 1.

average firewalled peer to obtain a fair sharing ratio, as the firewalled peers as a group have to download more than they can upload. The distribution of the upload burden over the firewalled and connectable peers determines the exact sharing ratio for each firewalled peer individually.

The bound on the amount of data the firewalled peers are able to upload results in bounds on the average sharing ratio for both firewalled and connectable peers. We shall not regard the injector's sharing ratio, for the following reasons. The injector only uploads data, giving it a sharing ratio of infinity. Also, some peers will not be able to upload any data (for example, consider a system with one injector and one downloader). These imbalances are expected to average out once the peers have downloaded multiple files. To keep the sharing ratio of the injector finite when considering a single file, we will assume that the injector downloads the file as well. This introduces a relative error in the order of $1/N$ in all the derived bounds, since the file will be downloaded by the injector superfluously. The following lemma gives bounds on the sharing ratios of the firewalled and the connectable peers.

Lemma 2. *For the average sharing ratios of firewalled and connectable peers, we have $S_{\mathcal{F}} \le (1/f) - 1$, and $1/(1 - f) - 1 \le S_{\mathcal{C}} \le 1/(1 - f)$, respectively.*

Proof. Using Lemma 1, the average debt for the firewalled peers is $\Delta(\mathcal{F})/F \ge (2 - 1/f - 1/(f \cdot N))L$, which converges to $(2 - 1/f)L$ for large N. Since every peer downloads exactly L bytes, the average sharing ratio of the firewalled peers is thus $S_{\mathcal{F}} = (L - \Delta(F)/F)/L \le 1/f - 1$.

The connectable peers can be shown to have a sharing ratio if at least $1/(1 - f) - 1$ on average using the same method. The average sharing ratio of the connectable peers is also bounded from above. Together, the N peers download $N \cdot L$ bytes. If all these bytes are uploaded by connectable peers, then the average sharing ratio for the connectable peers is $S_{\mathcal{C}} = N \cdot L/(C \cdot L) = 1/(1-f)$, otherwise it is smaller. □

The lower bound for the firewalled peers and the upper bound for the connectable peers are related. Every peer downloads the file once and has to upload it once, on average. As a result, the average sharing ratio of all peers is

1. However, for a subset of the peers this average does not necessarily hold.

Figure 2 plots the bounds of Lemma 2 against the fraction of firewalled peers f. These bounds are only met if all connectable peers obtain their data from the firewalled peers. In the case of $f < 0.5$, the bounds of Lemma 2 still hold, but the average sharing ratio will be closer to 1 if the data distribution algorithm does not force all data for connectable peers to originate from firewalled peers.

4.2 Practical Implications

The bounds derived so far depend only on the connectivity of the peers in each swarm. The bounds even hold if a peer seeds (uploads the file to others after completing the download) in an attempt to restore its sharing ratio. The upload speeds of the peers are irrelevant as well, as only the amount of data is considered. Furthermore, the results hold for file downloading as well as video-on-demand, and can be trivially extended to live video streaming. For the latter, the duration a peer decides to watch the video stream has to be taken into account, but that duration can be assumed to be independent of the connectivity of the peer. Finally, our results hold regardless of the data distribution method used, and therefore cover swarm-based distribution methods (which we consider in this paper) as well as tree-based distribution methods, as long as data is exchanged using unicast connections.

If $f > 0.5 + 0.5/N$, the connectable peers have to upload more than the firewalled peers. The injector could find this undesirable, and aim at an equal sharing ratio for all peers
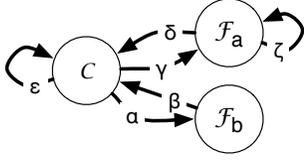
Figure 3. Possible data flows between the connectable peers $\mathcal{C}$ and firewalled peers $\mathcal{F}_a$ and $\mathcal{F}_b$ (puncturable and non-puncturable, respectively).

instead. Such a situation can be obtained if the injector increases its upload capacity to serve the firewalled peers in order to lower the sharing ratios of the connectable peers to the level of the firewalled peers. Although the average sharing ratio for both sets of peers will be below 1, fairness is nevertheless achieved as both sets have an equal uploading burden. Using Lemma 2, a lower bound on the upload capacity for the injector can be derived as follows.

Theorem 1. *It is impossible for all peers to obtain an equal sharing ratio if the injector uploads less than $(2-1/f)N \cdot L$ bytes to the firewalled peers.*

Proof. To create an equal sharing ratio for all peers, the connectable peers have to upload at least $S_\mathcal{C} \cdot L - S_\mathcal{F} \cdot L$ bytes less on average. Using Lemma 2, for all connectable peers combined, this amounts to at least $C \cdot (1/(1-f) - 1 - (1/f-1))L = (2-1/f)N \cdot L$ bytes. These bytes have to be uploaded by the injector instead. $\square$

If the fraction of firewalled peers f can be measured or estimated, the injector can thus predict a lower bound on the capacity needed to provide all downloaders with an equal sharing ratio. The actual required capacity is likely to be higher. If not all peers are on-line at the same time, some peers will be unable to upload data to each other. The injector will have to compensate for this if all peers are promised an equal sharing ratio.

5 Firewall Puncturing

In Section 2, we have discussed the general principles behind firewall puncturing. Whether a connection between two firewalled peers can be made depends on the types of firewall of both peers. There can be many different types of firewall for which a P2P distribution algorithm has implemented puncturing techniques. We will first analyse a system in which firewalls can be either punctured by everyone or not at all. We will then propose a more generic model.

Consider a system in which a firewall can be either punctured by all other peers, or by no one. Let $\mathcal{F}_a \subseteq \mathcal{F}$ be the set of peers with puncturable firewalls, and let $\mathcal{F}_b \equiv \mathcal{F} \setminus \mathcal{F}_a$ be the set of peers with non-puncturable firewalls. Also, define
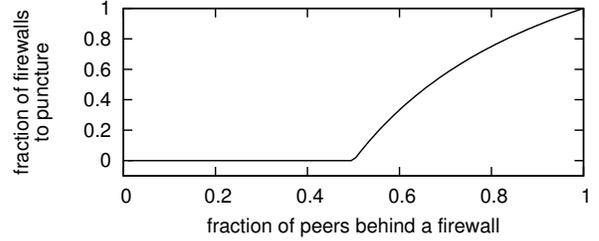

Figure 4. Lower bound on the fraction of firewalls that need to be punctured for a fair sharing ratio to be possible.

$F_a \equiv |\mathcal{F}_a|$ and $F_b \equiv |\mathcal{F}_b|$. Figure 3 shows the corresponding graph, where α to ε denote the total numbers of bytes transferred between the different groups of peers.

Theorem 2. *A fair sharing ratio for all peers is only possible if $F_a/F \geq 2 - 1/f$.*

Proof. We will assume $L = 1$ without loss of generality. To obtain a fair sharing ratio, peers in $\mathcal{F}_a$ and $\mathcal{F}_b$ have to upload $F_a + F_b = F$ bytes, so $\beta + \delta + \zeta = F$ has to hold. Peers in $\mathcal{C}$ download exactly C bytes, so $\beta + \delta + \varepsilon = C$, which leads to $\beta + \delta \leq C$. For the same reason, $\zeta \leq \zeta + \gamma = F_a$. Combining these equations yields $F \leq C + F_a$, or, equivalently, $f \cdot N \leq (1-f)N + F_a$, which leads to $F_a/N \geq 2f - 1$, and finally $F_a/F \geq 2 - 1/f$. $\square$

Figure 4 plots the lower bound of $2-1/f$ of the firewalls that need to be punctured against the fraction of firewalled peers f. For $f < 0.5$, a fair sharing ratio is (theoretically) possible without any firewall puncturing. The rapid rise in required firewall puncturing effectiveness is clearly visible once the fraction of firewalled peers is above 0.5.

The bounds in Lemmas 1–2 and Theorems 1–2 can be derived in a more general way by modelling a P2P system as follows. Let there be P types of firewall, of which one type represents having no firewall at all, and let P_i be the number of peers of firewall type i, $i = 1, \ldots, P$. Let $x_{ij} \geq 0$ be the number of bytes sent from peers of firewall type i to peers of firewall type j. Every peer has to download exactly L bytes, so

$$\sum_{j=1}^{P} x_{ij} = P_i \cdot L, i = 1, \ldots, P.$$

If a fair sharing ratio is desired, then all peers have to upload as many bytes as they download, resulting in

$$\sum_{j=1}^{P} x_{ji} = \sum_{j=1}^{P} x_{ij}, i = 1, \ldots, P.$$

By solving these equations for the x_{ij}, bounds can be derived for the average sharing ratio of the peers in each group.

As argued in Section 4.1, we have left out the injector, introducing small errors in the x_{ij}. To derive exact bounds, the injector can be added as a separate set of one peer. For $P = 2$, the model is consistent with the results derived in Section 4.

Whether a fair resource allocation can be obtained in a real setting depends on the P2P distribution algorithm as well as the arrival and departure pattern of the peers. After all, peers which are not on-line at the same time cannot connect to each other, regardless of firewalls or firewall puncturing techniques. The set of equations derived in this paper can be used to derive bounds on the supported fraction of firewalled peers in each group, or to check feasibility of an implementation.

6   Simulated Behaviour

We use a discrete-event BitTorrent simulator to evaluate how close the actual average sharing ratios for the firewalled and connectable peers are to the bounds derived in Section 4. We let 500 peers arrive (1 per second, on average) to download a 10 MByte file. A randomly chosen subset of the peers are firewalled. The injector is always on-line and is not firewalled. Each peer can upload with 0.5 Mbit/s to 0.75 Mbit/s, and has a download speed four times as high. The latency between each pair of peers is 100 to 300 ms.

We evaluate two departing policies. In the first policy, we let each peer depart if it completed the download and has obtained a fair sharing ratio as well. In the second policy, we run the same simulations, but let each peer depart directly when it has completed the download, regardless of its sharing ratio. We performed 180 simulations with each policy.

For each session, we record the number of firewalled and connectable peers, and sum the total amounts of data sent and received by both groups. The average sharing ratio of either group is then derived and shown as a dot in the subfigures of Figure 5. The theoretical bounds as derived in Section 4 are shown as well. Each measurement falls within the derived bounds. The figures for the two departure policies are similar. At low fractions of firewalled peers, the average sharing ratios of both groups tend towards 1, which is consistent with BitTorrent rewarding good uploaders with a good download performance. At high fractions of firewalled peers, the firewalled peers upload an amount close to their theoretical maximum. Almost all of them obtain the file from the injector, since other connectable peers are rare and quickly meet the departure requirements.

For fractions of firewalled peers smaller than 0.5, the average sharing ratios for the firewalled peers in Figures 5(a) and 5(b) differ slightly. If peers depart right after they complete their download, the connectable peers will have less opportunity to upload to each other, thus allowing the fire-

walled peers to upload to them more. This policy is interesting in systems which keep track of the sharing ratios of each peer across several sessions. In such systems, the firewalled peers could for example join sessions with a low fraction of firewalled peers, in order to increase a low sharing ratio. Both figures for the connectable peers show that high sharing ratios for them are realistic when the fraction of firewalled peers is high.

7   Behaviour of Real Systems

In order to assess the validity of the model we have presented in Sections 3, 4, and 5 in real systems, we have collected data on the behaviour of peers, and in particular, of firewalled peers and seeders, in several BitTorrent communities. Below, we first explain the difference between open and closed BitTorrent communities, then we discuss our two ways of collecting data on the behaviour of BitTorrent communities, and finally we present the results on the behaviour of these communities with respect to firewalled peers and seeders.

7.1   BitTorrent Communities

Every BitTorrent community uses one or more centralised servers (*trackers*) that keep track of the peers in each swarm; peers report to the (a) tracker when they join a swarm, and they report again when they leave. BitTorrent communities can be open or closed. Open communities use public trackers which anyone can join, and they have no explicit mechanisms for sharing-ratio enforcement. Closed communities use private trackers to prevent outsiders from joining their swarms. In closed communities, every user has an account on a centralised server and sharing ratios are enforced by banning peers whose sharing ratios drop below a (secret) treshhold. Peers are expected to seed content in order to maintain or restore their sharing ratios. Unfortunately, we know of no BitTorrent community that publishes the sharing ratios of the individual peers, making it impossible to verify the bounds derived in this paper directly. We can therefore offer only correlations between our theory and the measured results.

7.2   Data Collection

We have employed two methods for collecting data on the behaviour of BitTorrent communities, one in which we monitor a community for a period of time, and one in which we take snapshots of communities. As to the first method, we use a data set that we have collected in May 2005 [11], which consists of monitoring information of the operation of 1,995 BitTorrent swarms during one week in the (open) The Pirate Bay community.

We extend the analysis presented in [11] by extracting the behaviour of the firewalled peers from this data set. During the data collection, every two minutes, all peers that

(a) Peers depart at a fair sharing ratio.

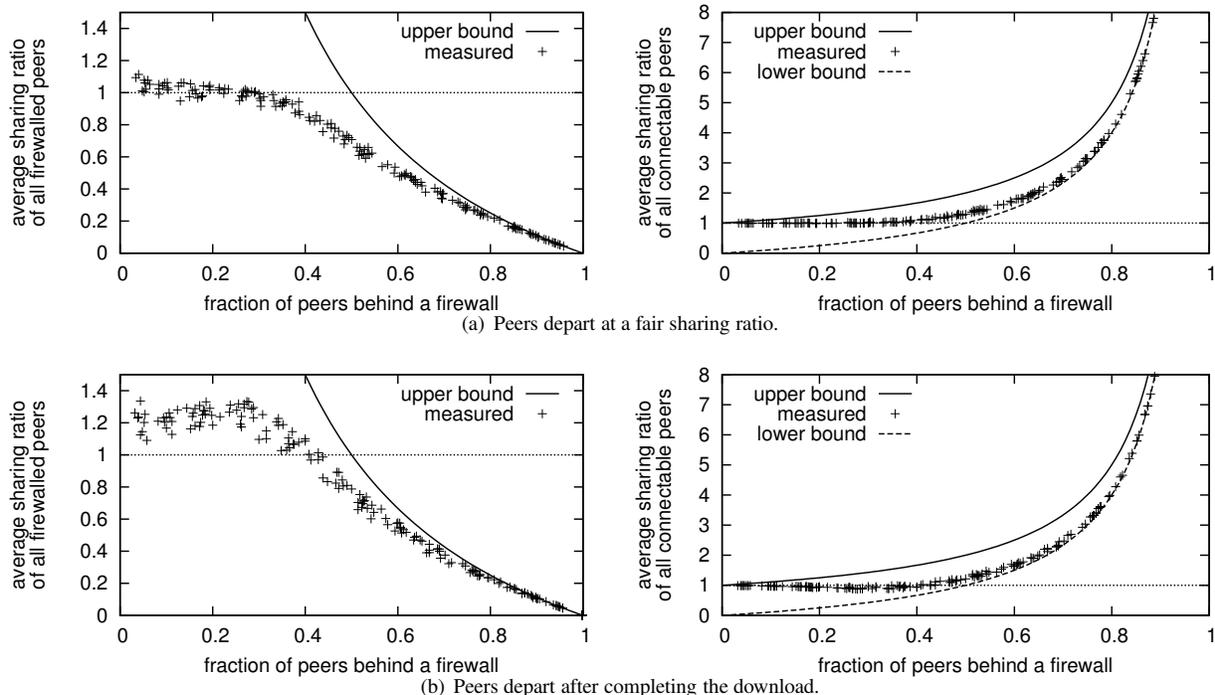(b) Peers depart after completing the download.

Figure 5. The average sharing ratios for the firewalled peers and the connectable peers each session for two departure policies. The horizontal dotted lines represent a fair sharing ratio of 1.

Nr	Community	Type	Swarms	Source
1	TheBox	closed	2,430	`thebox.bz`
2	TVTorrents	closed	2,363	`tvtorrents.com`
3	BTJunkie	closed	3,267	`btjunkie.org`
4	The Pirate Bay	open	7,368	`thepiratebay.org`
5	BTJunkie	open	16,400	`btjunkie.org`

Table 1. The communities for which we have used the second method of data collection. BTJunkie actually collects statistics on both open and closed communities.

were reported to exist in each of the existing swarms were contacted. We consider a peer to be firewalled if it is repeatedly reported to exist but could never be contacted.

As to the second method of collecting data, we have taken snapshots of several BitTorrent communities in January 2008, which are listed in Table 1. These communities publish on their web sites the number of downloaders and seeders in each swarm, and we have collected these statistics at a single instant in time for one type of content (TV shows), because it was present in all communities. Nevertheless, we do consider the measurements to be representative as they contain swarms of varying ages. Of the communities in Table 1, TVTorrents also publishes which peers are actually present in which swarm, and whether they are

firewalled or not. We collected these data for 557 swarms.

7.3 Behaviour of Firewalled Peers

In this section we present the behaviour of the The Pirate Bay (using data collected with our first method) and of TVTorrents (using the second method) with respect to firewalled peers. The TVTorrents data is used in Figure 6 (top) only. As can be seen in Figure 6 (top), which plots the cumulative distribution function (CDF) of the percentage of (unique) firewalled peers within each swarm, a substantial fraction of the peers in each swarm is firewalled. For the The Pirate Bay, the average swarm has 66% of its peers firewalled, and more than 96% of the swarms have more than half of the peers behind a firewall. In contrast, the average swarm in the closed TVTorrents community only has 45% of its peers behind a firewall, and 24% of the swarms have more than half of the peers behind a firewall. The reason for the difference between these communities is beyond our abilities to measure, but the difference does correlate with our theory. Even though the measurements of both communities were taken 20 months apart from each other, measurements by others do not suggest a decrease in the fraction of firewalled peers over time [2, 5, 6, 17, 21–23]. We believe this difference to be due to the policy of banning peers with low sharing ratios in closed communities such as TVTor-
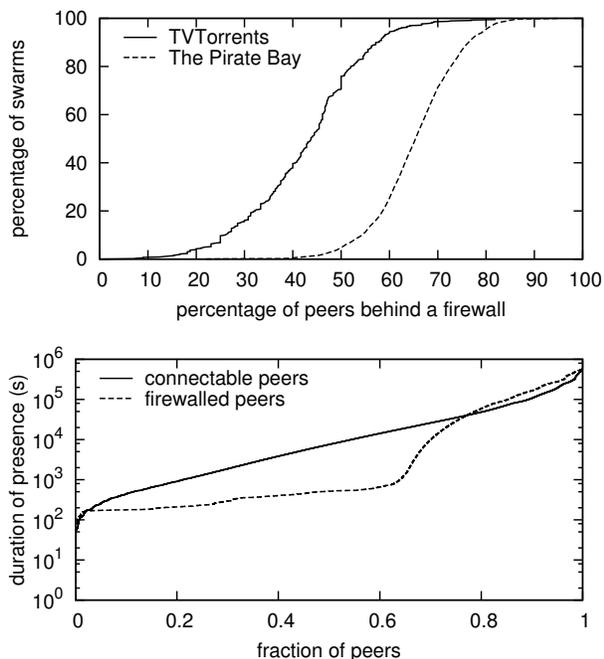
Figure 6. The CDF of the percentage of fire-walled peers per swarm (top) and of the durations of the presence of the peers (bottom).
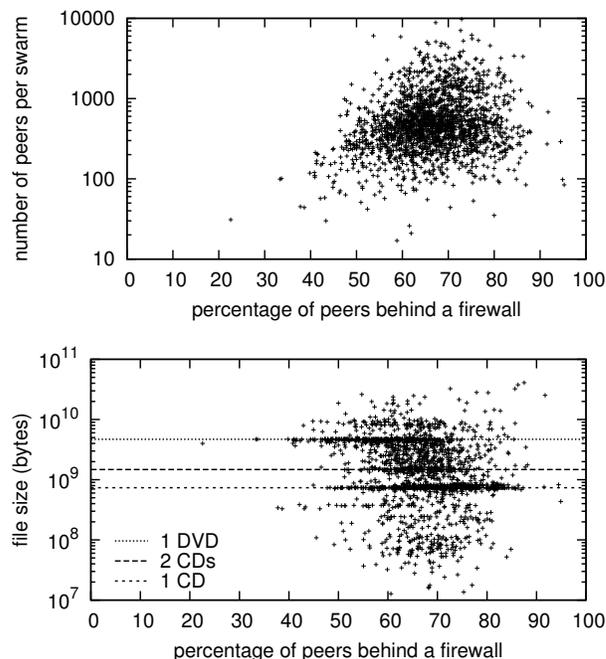
Figure 7. A scatter plot of the number of peers per swarm (top) and of the size of the file being downloaded (bottom) versus the percentage of firewalled peers.

rents. First, our analysis in Section 4 shows that firewalled peers have more difficulty obtaining a fair sharing ratio, and so they have a higher probability of getting banned. Secondly, to reduce the risk of getting banned, a firewalled peer has an incentive to configure its firewall to accept incoming connections. A closed community thus favours users which have the technical knowledge to do so.

The BitTorrent protocol has been designed to approximate a fair sharing ratio by giving the best performance to peers that upload fastest. A low upload speed results, on average, in a low download speed. As a consequence, the firewalled peers will have a harder time finding connectable peers to upload to, often resulting in poor performance. Also, the firewalled peers do not necessarily have the same upload capacity as the connectable peers.

Figure 6 (bottom) plots how long the firewalled and connectable peers stayed in the system, measured as the difference between the first and the last time they were contacted. The firewalled peers are biased towards both a short and a long presence. A short presence can be explained by firewalled peers not being able to contact enough connectable peers at start-up, resulting in poor performance, after which the download is aborted by the user. When performance is acceptable, the firewalled peers stay in the system longer than the connectable peers. A possible explanation is that the firewalled peers have to complete their download with a lower download speed, again due to connectivity prob-

lems. On average, the presence of a firewalled peer was 12.8 hours, while the presence of a connectable peer was only 10.6 hours. For the peers that stayed for at least an hour, these averages rise to 38.5 and 17.1 hours, respectively. We found that at any moment, 72% of the on-line peers were firewalled on average, due to their longer presence in the system.

The distribution of the fraction of firewalled peers f as shown in Figure 6 (top) does not suffice to assess the actual impact of firewalled peers, since it does not provide insight into the absolute number of peers and the size of the file that is shared in the swarms with $f > 0.5$. Therefore, we have evaluated the correlations between the fraction of firewalled peers and the swarm size, as well as the size of the shared file. Figure 7 (top) is a scatter plot of the number of unique peers in a swarm versus the percentage of firewalled peers in the swarm. There is no strong correlation (the correlation coefficient is 0.22), indicating that popular files (swarms with many peers) do not seem to have a significant bias in the percentage of firewalled peers. Figure 7 (bottom) shows the size of the file exchanged within a swarm against the percentage of firewalled peers in the swarm (the correlation coefficient is -0.26). There is a bias in file size as many swarms exchange files with a size corresponding to 1 or 2 CDs (703 Mbyte per CD) or to 1 DVD (4.37 GByte). This bias is not surprising as many of the
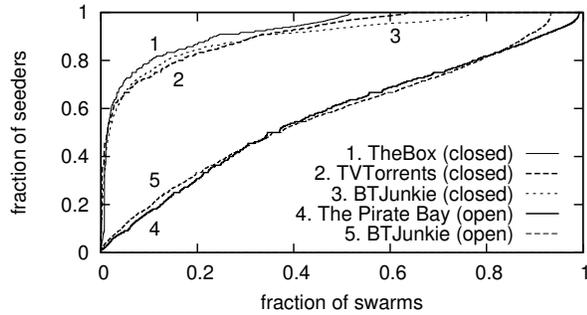
Figure 8. The fraction of peers which are seeding for open and closed communities.

files were actually movies, transcoded to fit on either media. The swarms exchanging DVDs contain a slightly smaller percentage of firewalled peers compared to the swarms exchanging CDs. We conjecture that one of the reasons for this phenomenon is that many users have ADSL connections. On the one hand, such connections are (relatively) slow, leading to a preference for CD versions of movies, and on the other hand, such connections typically employ a NAT-router.

7.4   Fraction of Seeders

In this section we present the behaviour of the BitTorrent communities listed in Table 1 (using data collected with our second method) with respect to seeders. Figure 8 shows the CDF of the fraction of seeders in the swarms at the time of snapshot. The numbers for BTJunkie are split for their statistics on open and closed communities.

Open communities do not keep track of their users across swarms, so there is no reason for a peer to stay on-line to seed in a swarm after the download has finished. Some users will nevertheless seed, out of altruism, or because their client is configured to do so. A common default configuration for BitTorrent clients is to seed until the user aborts the program, or until a fair sharing ratio is reached for that swarm.

In the closed communities, swarms have higher fractions of seeders, and there is a higher fraction of swarms which contained only seeders at the moment of measurement. Once a peer joins such a swarm, it will experience a high download speed due to the many seeders present, but it will also have to compete with these seeders to restore its sharing ratio. The peers are forced by the sharing ratio enforcement to nevertheless try to restore their ratio by seeding as much content as possible. The firewalled peers will have to seed longer on average, as they can only connect to connectable arriving peers. The latter is a possible explanation for the many swarms in which seeds are idle, waiting for a peer to arrive and to start downloading from them.

Section 7.3 has shown that the average swarm has almost

half its peers behind a firewall, implying that the distribution of the file data needs to be near-perfect if a fair sharing ratio is to be possible for the average firewalled peer. In theory, closed communities can optimise the communication between peers using sharing ratio information, but in practice, they do not. We expect the data distribution to be closer to our simulations than to a near-perfect distribution. A fraction of the firewalled peers will not be able to obtain a fair sharing ratio, and thus effectively will have to seed.

8   Related Work

To the best of our knowledge, we are the first to derive theoretical bounds on the sharing ratio that can be achieved in P2P networks. Algorithms designed for P2P data distribution generally do not take firewalls into account, even those that are designed with fairness in mind [10, 12, 13, 15, 16, 20]. Ripeanu et al. [18] confirm some of our findings and recognise that the limited connectivity of firewalled peers makes it harder for them to contribute. Also, the high fraction of seeders in closed communities has been recognised by them and others [3, 18]. However, neither study looked at the fundamental limits imposed by firewalls and the resulting necessity in closed communities of having many seeders in many swarms.

The percentages of firewalled peers we measured are consistent with previous measurements [2, 5, 6, 17, 21–23], but these vary widely in their results (between 25% and 93%). Firewall puncturing and NAT traversal techniques have been researched [4, 7–9, 19], and proven to be quite effective for the considered sets of firewalls and NATs. However, in practice, these techniques are non-trivial to implement and maintain, and often need a third (connectable) party to coordinate the puncturing between two firewalled peers. Also, for best results, one is forced to use UDP [19] instead of TCP, as the latter requires more complex puncturing techniques which depend on the types of firewall present in the network [4, 7, 8].

9   Conclusions

We have shown that in P2P file-sharing networks, there is a trade-off between security and fairness: peers behind firewalls do not accept incoming connections, and as a consequence, if there are too many of them, there is no way freeriding can be prevented. We have provided bounds on the sharing ratios that can be obtained by firewalled and connectable peers, which hold regardless of how long firewalled peer remain in the system to seed to other peers, and regardless of their connection speeds. Firewall puncturing and NAT traversal techniques may alleviate the freeriding problem, and we provided a lower bound on the fraction of

firewalls that have to be punctured to be able to obtain a fair sharing ratio. Because firewall and NAT techniques evolve, a standard for both firewall puncturing and NAT traversal is required to guarantee fairness in P2P networks.

We have both run simulations and real-world measurements to validate our theory. In our simulations, we have shown the actual behaviour of the average sharing ratios of firewalled and connectable peers. When almost all peers are connectable, the BitTorrent protocol is sufficient to keep the system fair, as both groups have an average sharing ratio close to 1. The average sharing ratio of the firewalled peers decreases and converges to the theoretical upper bound as the fraction of firewalled peers increases. We have also collected and analyzed data on the behaviour of both open and closed P2P communities which reveal the real-world behaviour of firewalled peers. It turns out that the fraction of firewalled peers is significant: In closed communities, 45% of the peers are firewalled, versus 66% in open communities. A peer can increase its sharing ratio by seeding, and indeed we found a significantly higher fraction of seeders in the closed communities, indicating that a fair sharing ratio is not easy to obtain. The observed behaviour is consistent with our analysis, and can aid in the design of new P2P data-distribution algorithms.

References

[1] E. Adar and B. Huberman. Freeriding on Gnutella. *First Monday*, 5(10), 2000.

[2] S. Agarwal, J. P. Singh, A. Mavlankar, P. Baccichet, and B. Girod. Performance and Quality-of-Service Analysis of a Live P2P Video Multicast Session on the Internet. In *Proc. of the 16th IEEE Int. Workshop on Quality of Service (IwQoS)*, 2008.

[3] N. Andrade, M. Mowbray, A. Lima, G. Wagner, and M. Ripeanu. Influences on Cooperation in BitTorrent Communities. In *Proc. of ACM SIGCOMM*, pages 111–115, 2005.

[4] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig. NAT-BLASTER: Establishing TCP Connections Between Hosts Behind NATs. In *Proc. of ACM SIGCOMM Asia Workshop*, 2005.

[5] H. Burch and D. Song. A Security Study of the Internet: An Analysis of Firewall Behavior and Anonymous DNS. Technical Report CMU-CS-04-141, Carnegie Mellon University, 2004.

[6] Y. Chu, A. Ganjam, and T. Ng. Early Experience with an Internet Broadcast System Based on Overlay Multicast. In *Proc. of USENIX*, 2004.

[7] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *Proc. of USENIX*, page 13, 2005.

[8] S. Guha and P. Francis. Characterization and Measurement of TCP Traversal Through NATs and Firewalls. In *Proc. of the Internet Measurement Conference (IMC)*, pages 199–211, 2005.

[9] S. Guha, Y. Takeda, and P. Francis. NUTSS: A SIP-based Approach to UDP and TCP Network Connectivity. In *Proc. of ACM SIGCOMM workshop on Future directions in network architecture*, pages 43–48, 2004.

[10] M. Haridasan, I. Jansch-Porto, and R. van Renesse. Enforcing Fairness in a Live-Streaming System. In *Proc. of SPIE, Multimedia Computing and Networking Conference (MMCN)*, volume 6818, Article 68180E, 2008.

[11] A. Iosup, P. Garbacki, J. Pouwelse, and D. Epema. Correlating Topology and Path Characteristics of Overlay Networks and the Internet. In *Proc. of the 6th Int. Workshop on Global and Peer-to-Peer Computing*, 2006.

[12] H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR Gossip. In *Proc. of the 7th USENIX Operating Systems Design and Implementation (OSDI)*, pages 191–206, 2006.

[13] Q. Lian, Y. Peng, M. Yang, Z. Zhang, Y. Dai, and X. Li. Robust Incentives via Multi-level Tit-for-tat. In *The 5th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[14] Microsoft Corporation. Universal Plug and Play Internet Gateway Device v1.01, 2001.

[15] J. Mol, D. Epema, and H. Sips. The Orchard Algorithm: Building Multicast Trees for P2P Video Multicasting Without Free-riding. *IEEE Transactions on Multimedia*, 9(8):1593–1604, 2007.

[16] J. Mol, J. Pouwelse, M. Meulpolder, D. Epema, and H. Sips. Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems. In *Proc. of SPIE, Multimedia Computing and Networking Conference (MMCN)*, volume 6818, Article 681804, 2008.

[17] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The BitTorrent P2P File-Sharing System: Measurements and Analysis. In *Proc. of the 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 205–216, 2005.

[18] M. Ripeanu, M. Mowbray, N. Andrade, and A. Lima. Gifting technologies: A BitTorrent case study. *First Monday*, 11(11), 2006.

[19] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN-Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489, 2003.

[20] V. Venkataraman, P. Francis, and J. Calandrino. Chunkyspread: Multi-tree Unstructured Peer-to-Peer Multicast. In *Proc. of the 5th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.

[21] W. Wang, H. Chang, A. Zeitoun, and S. Jamin. Characterizing Guarded Hosts in Peer-to-Peer File Sharing Systems. In *Proc. of IEEE GLOBECOM*, volume 3, pages 1539–1543, 2004.

[22] S. Xie, G. Y. Keung, and B. Li. A Measurement of a Large-Scale Peer-to-Peer Live Video Streaming System. In *Proc. of the IEEE Intl. Conf. on Parallel Processing Workshops*, page 57, 2007.

[23] M. Yang, Z. Zhang, X. Li, and Y. Dai. An Empirical Study of Free-Riding Behavior in the Maze P2P File-Sharing System. In *Proc. of the 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 182–192, 2005.