Efficiency of Data Distribution in BitTorrent-Like Systems

Ho-Leung Chan[1,⋆], Tak-Wah Lam[2], and Prudence W.H. Wong[3]

[1] Department of Computer Science, University of Pittsburgh
hlchan@cs.pitt.edu
[2] Department of Computer Science, University of Hong Kong, Hong Kong
twlam@cs.hku.hk
[3] Department of Computer Science, University of Liverpool, UK
pwong@csc.liv.ac.uk

Abstract. BitTorrent (BT) in practice is a very efficient method to share data over a network of clients. In this paper we extend the recent work of Arthur and Panigrahy [1] on modelling the distribution of individual data blocks in BT systems, aiming at a better understanding of why BT can achieve a high degree of parallelism. In particular, we include in our study several new network features that BT systems are using, as well as different local heuristics for routing data blocks in each client. We conduct simulations to figure out to what extent the new network features and routing heuristics would affect the distribution efficiency. Our findings confirm that for the primitive network setting studied in [1], it does require $\Omega(b \log n)$ phases for n clients to download b data blocks. More interestingly, our work suggests that for the more realistic network setting, the heuristics Random and Rarest Block First both allow n clients to download b blocks in $b + O(\log n)$ phases. We believe that the latter bound better reflects the high degree of parallelism of BT observed in reality. It is also worth-mentioning that $b + \log n$ is the smallest possible number of phases needed; it is interesting to see that some simple local routing heuristics have a performance so close to the optimal.

1 Introduction

Let us consider the following problem. There are n clients (nodes) on a well-connected network. They want to download a file of b data blocks from a server in a cooperative and efficient, but distributed manner. The idea is to avoid each client directly downloading from the server. Instead the server uploads each data block to only one client, and let the clients distribute the block among themselves. Assume each client can upload one data block to only one neighbor in one phase. The key concern is whether there exists a good strategy for each client to determine in each phase which block and which neighbor to upload, so that most clients can have progress in each phase.

⋆ Part of this research was done while the author was at University of Hong Kong.

The above problem is based on the "flash crowd" scenario, where a large number of clients join the network almost simultaneously to download a data file (say, a soccer game). BitTorrent (BT) [3,7] has found to be a very efficient method for such a problem in practice; it does exploit the bandwidth among the clients, and it is often observed that using BT, most clients have swift progress in parallel. Arthur and Panigrahy [1] were the first to model the distribution of data blocks in BT systems mathematically, aiming at explaining the high degree of parallelism BT achieves. In particular, they consider the clients are connected via a directed BT-graph (definition given in Section 2), and in each phase each client can upload (send) as well as download (receive) one data block from its neighbors. Among others, they proved that using a random strategy, the n clients can download all b blocks in $O(b \log n)$ phases with high probability. Below we refer this number of phases as the *total distribution time*.

In this paper, we extend the model used in [1] to include other network features found in BT networks. First, traffic between two neighbors is usually bidirectional; i.e., a client can upload and download from each neighbor client. Next, we note that the size of a data block is chosen in such a way that in each phase, a client has sufficient bandwidth to upload a block to one of its neighbors, yet the download rate is usually a few times higher than the upload rate (this is imposed by some internet providers), and each client can receive several data blocks in each phase. Furthermore, each client should be able to make a better decision using a request-based protocol instead of a push-based protocol; the former protocol requires each client to listen to the requests from its neighbors before making a decision which block and which neighbor to upload. The first objective of this paper is to study to what extent these extended features of the model would affect the total distribution time.

Furthermore, we study three different local routing heuristics for each node to decide which block to upload to its neighbor in each phase, namely, Sequential, Random, and Rarest Block First. Sequential simply uploads blocks sequentially according to their order in the file. Random picks a random block to upload. Rarest Block First selects a block which is the rarest among the neighbors of a node; this is a strategy being used by BT [5,7]. Regardless of which heuristic a node uses, the node only uploads a block which the neighbor does not have.

Note that no matter which network setting and which heuristics we pick, the total distribution time is at least $b + \lceil \log n \rceil$. To see this lower bound, we observe that the server requires b phases to upload b blocks, and the last block can reach at most $2^i - 1$ clients after i phases. It is a challenging task to find a reasonable network setting and routing heuristic that can lead to a total distribution time close to $b + \log n$.

We have done a lot of simulation on different network settings and heuristics. We consider models with different combinations of the three network features mentioned above to see to what extent they affect the total distribution time. These features include directed vs undirected graph, single vs multiple receive, and push vs request protocol. For multiple receive, we further distinguish the cases where a limited number x (called receive-x) of blocks vs an unlimited number can be received by a node in each phase. For each combination, we study the three local routing heuristics. We also vary the number of nodes n and the number of blocks b. Our findings are summarized as follows.

– As expected, the undirected and request-based model admits a better data distribution.
– For the model used in [1] (i.e., directed_receive-one_push), we observe that the total distribution time does require $\Omega(b \log n)$.
– The most surprising result is related to the model using undirected BT-graph, multiple receives, and request-based protocol (i.e., undirected_receive-x_request). No matter random or rarest block first is used, our simulation shows that the total distribution time is in the order $b + O(\log n)$, very close to the lower bound. For example, when $b = 3000$ and $n = 2000$, the total distribution time is around 3050 on average (with a very small standard deviation of 5 over five different trails.)
– It is natural to expect that the total distribution time to decrease with maximum number of receives in each phase increases. The decrease is indeed very drastic when we vary the number from 1 to 2, but the effect is not visible once the number goes up to 3. In fact, we find no significant difference using a maximum of five receives and an unlimited number of receives.
– Rarest block first is a heuristic used by BT. It is most effective, but only a bit better than Random. On the other hand, using Random saves a lot of implementation overhead. We believe Random is a better choice.

Related work. Before the work of Arthur and Panigrahy [1], there have been some theoretical work, yet some assumptions were made which might not be true in practice. Qiu and Srikant [10] applied flow analysis to BT-like networks but assuming (1) the data blocks available in each client for download at each phase is random and independent; (2) constant arrival rates. The latter does not account for BT's strength in handling flash crowd scenario. Yang and de Vecianna [11] considered flash crowd scenario but assumed that distribution of one data block will not slow down the distribution of other data blocks, ignoring the possible interaction in the distribution. There is also empirical work attempting to demonstrate BT's routing policy work well in practice [8,9,2,6], but they do not consider the distribution of individual data block.

Organization of the paper. In Section 2, we review Arthur and Panigrahy's model and discuss the effects of various network features on data distribution time. In Section 3, the local routing heuristics are tested. Finally, we give some concluding remarks in Section 4.

2 Network Models and Distribution Time

In this section, we first review Arthur and Panigrahy's model and then discuss how different model features affect the total distribution time. These features include directed vs undirected graph, single vs multiple receive, and push vs request protocol. In the next section we will study the effect of different routing heuristics. The experiments in this section all assume the Random heuristic when a node decide which block to send/receive.

2.1 Arthur and Panigrahy's Model

Arthur and Panigrahy [1] model a BT-like network as a directed graph with n nodes, each representing a user. One of the nodes is the seed which holds a large file initially. The file is divided into b equal-size blocks. The remaining $n - 1$ nodes want to obtain the file and they have zero block of the file initially.

A BT system maintains a virtual network represented as a BT-C graph[1], for some integer C, which is constructed as follows. The BT-C graph starts with C nodes $v_1, v_2, \ldots, v_C$, with a directed edge from v_i to v_j if and only if $i < j$. While the total number of nodes is less than n, a new node is added. C existing nodes are selected at random from which directed edges are drawn to the new node. In most BT systems, C is set to 40 .

For the distribution of data blocks in the network, a node can send a block to a neighbor only if it already has the block, and a node obtains the whole file only if it has every block of the file. Time is divided into discrete time steps (*phases*). In each phase, each node can send at most one block to a neighbor along an outgoing edge. When multiple blocks are sent to a node in a phase, the node can receive at most one such block.

The efficiency of the system is measured by the *total distribution time*, which is the number of phases taken until all nodes obtain the file. Arthur and Panigrahy assume a simple protocol for sending blocks: In each phase, each node u randomly picks a neighbor v, and u sends to v a block that u already has but v does not. u is idle if no such block exists. They proved that the total distribution time is $O(b \log n)$ with high probability.

2.2 Identifying More Realistic Model Features

We observe that some features in Arthur and Panigrahy's model are too restrictive comparing to a real BT system. In particular, we focus on the following three important features and analyze how they affect the total distribution time.

1. **Directed vs Undirected graph.** When modelling the network as a directed graph, the connection is asymmetric, i.e., there are pairs of nodes, say u and v, connected by an edge for which u can send to v but not vice versa This is not the case in real-life BT systems (more precisely, the underlying Internet) in which connected nodes can send blocks in both directions. In our study, we analyze the effect of assuming directed vs undirected (bidirectional) edges.
2. **Single vs multiple receive.** This models the bandwidth limit of the nodes. When multiple blocks are sent to a node in a phase, Arthur and Panigrahy assume that the node randomly obtains one of the blocks. It corresponds to

[1] The network in which the users connect is actually the Internet. On the application layer, BT maintains a virtue network in the form of a BT-C graph.

Table 1. The most restrictive model DG-1-Ph (col. 1) performs the worst. The most relaxed models UG-5-Rq & UG-u-Rq (bolded) are the best. Request protocol improves distribution time (col. 3 vs 4; 5 vs 6). Multiple receive also outperforms single (col. 2 vs 4 & 6).

Distribution time	DG-1-Ph	UG-1-Rq	UG-5-Ph	**UG-5-Rq**	UG-u-Ph	**UG-u-Rq**
$n = 300, b = 1200$	3826	1913	1354	**1228**	1325	**1228**
$n = 2000, b = 1200$	5068	1935	1386	**1247**	1344	**1239**
$n = 300, b = 3000$	9915	4742	3300	**3034**	3212	**3030**
$n = 2000, b = 3000$	12866	4787	3368	**3047**	3243	**3044**

the situation that download bandwidth is limited to a similar extent as the upload bandwidth. In reality, the download bandwidth of a node is usually higher than the upload capacity. Thus, it is interesting to understand the effect when each node can receive $r \geq 2$ blocks in each phase. We will consider the cases of $r = 1$ vs $r = 5$ and r is unlimited.

3. **Push vs request protocol.** This refers to whether a node actively ask for a *missing* block that it does not have. When a node send blocks to the neighbors, it was assumed in [1] that the node randomly picks a neighbor and sends a block useful to that neighbor without prior communication. We call this a *push* protocol. In reality, a BT system uses a two-way protocol where at the beginning of each phase, each node sends requests to the neighbors asking for missing blocks, and then each node serves some of the requests it gets. We call this a *request* protocol. Intuitively, request protocol helps to avoid multiple neighbors sending the same block to a node.

2.3 Simulation and Findings

We consider models with different combinations of the above three features and see how they affect the total distribution time. We name the models using three fields: the first field is either DG (directed graph) or UG (undirected graph); the second field tells the maximum number of blocks a node can receive in one phase, with u means unlimited; the third field is Ph (push protocol) or Rq (request protocol). For example, DG-1-Ph refers to the model studied in [1]. On the other hand, UG-u-Rq is the most relaxed model, based on which we vary the features to also study models in between (see Table 1 for the list of models we study).

When the request protocol is used, we assume that a random routing heuristic is used, i.e., each node picks a random missing block in turn (until all missing blocks have been considered) and requests it from a random neighbor having it, with the restriction that no neighbor will be requested twice. Note that we will discuss other routing heuristics in Section 3. We also assume that after getting the requests, each node randomly serves one of the requests.

We perform simulations with $n = 300, 2000$ and $b = 1200, 3000$. Note that BT systems usually divide a file into pieces of 1/4 MB each, $b = 1200$ and 3000 correspond to a file of 300MB and 700MB, respectively, the latter is about the

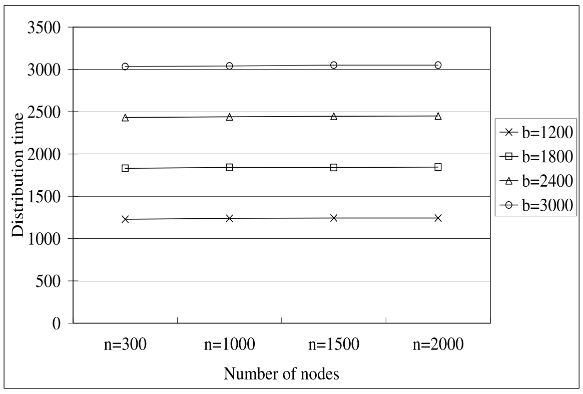

Fig. 1. The distribution time increases linearly with b (see the four different curves). The effect of n on the distribution time is small (see the flatness of each curve).

size of a CD. The choices of n allow us to see the effect of n being smaller than and larger than b. For each model and combination of n and b, we repeat the simulation for 5 times and record the average distribution time. The results are shown in Table 1. We have the following observations.

Near optimal distribution time. We observe that the total distribution time of models UG-5-Rq (col. 4) and UG-u-Rq (col. 6) are close to the theoretical lower bound of $b + O(\log n)$. It shows that the features of undirected graph, multiple receive and request protocols are very effective in improving the total distribution time. Note that a random heuristic is used in selecting blocks to send and receive. The experiments show that this is sufficient to obtain a good performance under an appropriate model.

To understand further the growth in distribution time under different combinations of n and b, we perform more experiments under the UG-5-Rq model, with $n = 300$, 1000, 1500, 2000 and $b = 1200, 1800, 2400, 3000$. We plot the total distribution time against n, with each value of b in a different curve. The results are shown in Figure 1. We can observe that the total distribution time grows very slowly as n increases. The distribution time is close to $b + k \log n$ where k is approximately 4.

Multiple receive is most effective. We observe that among the three features considered, allowing multiple receive per phase seems to be the most effective in improving the total distribution time. For example, comparing the models of UG-1-Rq (col. 2) and UG-5-Rq (col. 4), the total distribution time decreases by about 35%.

The effect of multiple receive can be explained as follows. Because each node decides on its own to whom it sends a block, it is common for multiple nodes sending blocks to the same target node. When each node can only receive one block per phase, bandwidth is lost, i.e., overall the number of data blocks received is smaller than that sent because some blocks have to be dropped. We call this

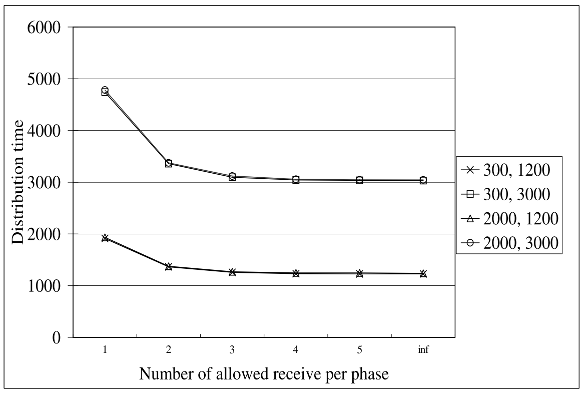

Fig. 2. The total distribution time drops significantly when the number of allowed receive increases from 1 to 2 and becomes stable beyond 3. Note that the two curves for $b = 1200$ (similarly, for $b = 3000$) are very close because the distribution time is dominated by the value of b.

receive collision. Multiple receive effectively reduces the bandwidth lost due to receive collision, thus leading to improved distribution time.

To obtain a better understanding of the effect of multiple receive, we vary the number of allowed receive from 1 to 5 and also unlimited. We perform experiments for $n = 300, 2000$ and $b = 1200, 3000$. The results are shown in Figure 2. We observe that the total distribution time improves greatly when number of allowed receive increases from 1 to 2, and has little effect beyond 3. It suggests that receive collision is common, but the number of blocks involved is usually small.

Undirected graph and request protocol are also effective. Undirected graphs have better distribution time than directed graphs (see col. 1 vs others) because of two reasons. Since the edges are bidirectional in an undirected graph, there are effectively double the possible connections among the nodes. Furthermore, the leaf nodes in directed graphs having no outgoing edges do not help to distribute the received blocks, so they reduce the efficiency by reducing the availability of the blocks.

Besides receive collision we mentioned before, there is another kind of collision that reduces the efficiency of distribution. Bandwidth is also lost when multiple copies of the same block are sent to the same node in a phase. We call this *send collision.* Request protocol avoids send collision because for each missing block, a node actively requests only one single neighbor to send the block.

Tightness of the analysis in [1]. Finally, we look at the results for the model DG-1-Ph [1] more closely. We observe that the total distribution time grows in the order of $b \log n$, it is roughly $0.4 \times b \log n$ (see col. 1). The experimental results suggest that data distribution does require $\Omega(b \log n)$ phases to finish in the DG-1-Ph model.

Table 2. Random and Rarest First have similar distribution time in the Model UG-5-Rq, which is close to the theoretical lower bound of $b + O(\log n)$

Distribution time	Random	Rarest First	Sequential
$n = 300, b = 1200$	**1228**	**1221**	3319
$n = 2000, b = 1200$	**1247**	**1233**	4678
$n = 300, b = 3000$	**3034**	**3024**	8494
$n = 2000, b = 3000$	**3047**	**3037**	11615

3 Performance of Simple Routing Heuristics

In BT systems, distribution of blocks is decided locally by each node, without a central coordination. Heuristics are used to determine which blocks are sent to which nodes at each phase. In this section, we consider both request and push protocols, and study how different heuristics affect the distribution time.

3.1 Routing Heuristics for Request Protocols

In models with request protocols, distribution of blocks is driven by what requests are sent by the nodes. We study three natural heuristics for deciding which blocks a node should request from its neighbors. To limit the communication overhead due to the requests, we restrict that at each phase, each node u can send at most one request for each block, and u can send at most one request to each of its neighbors.

1. **Random.** One simple heuristic for sending requests is by random. That is, at each phase, each node u repeatedly picks a random block it misses, and sends a request for it to a random unrequested neighbor having that block. This is the heuristic studied in the previous section, and it is observed that Random already achieves very good distribution time.
2. **Rarest First.** Real-life BT systems use the heuristic Rarest First to decide which blocks to request: At each phase, each node u counts the *availability* of each block it misses, where the availability of a block is the number of neighbors of u having that block. Then, starting from the rarest block (i.e., block with smallest availability), u sends requests for each block to a random unrequested neighbor having it.
   The motivation of Rarest First is to maintain balanced availability of each block in the network, so it avoids bad distribution time due to a small number of rare blocks.
3. **Sequential.** When a file is divided into blocks, each block corresponds to a different part of the file. With the heuristic Sequential, at each phase, each node sends requests for blocks sequentially according to their order in the file. The motivation for this heuristic is the ease of programming and it may lead to the good performance similar to that of Random. It may also support streaming of the file, i.e., the node can start using the file while the file is still being downloaded.

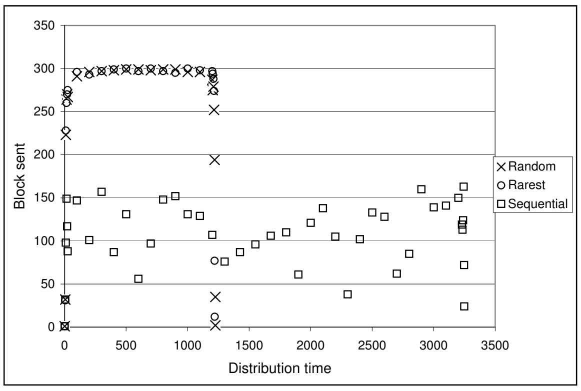

Fig. 3. The number of blocks sent in different phase during the file distribution, for $n = 300$ and $b = 1200$, in the Model UG-5-Rq. With Random and Rarest First, almost 300 blocks are sent per phase, while for Sequential, only 50 - 150 blocks are sent per phase.

Table 3. The distribution time of different heuristics in the Model UG-5-Ph. When a push protocol is used, Random has better distribution time than Rarest First in three out of the four cases (bolded).

	Random	Rarest First	Sequential
$n = 300, b = 1200$	**1354**	1356	8672
$n = 2000, b = 1200$	1386	1373	11365
$n = 300, b = 3000$	**3300**	3372	21594
$n = 2000, b = 3000$	**3386**	3407	28244

We assume that at each phase, each node randomly serves one of the requests it received. We study the performance of the three heuristics in the more realistic Model UG-5-Rq, i.e., undirected graph, at most 5 recevies per phase, and using request protocol. We perform simulations on different combinations of n and b. The results are shown in Table 2.

Random and Rarest First. We observe that the Random and Rarest First heuristics give very similar performance. Rarest First has slightly better distribution time than Random, but both can distribute the file in $b + O(\log n)$ phases.

To understand the reason for their efficiency, we investigate the case of $n = 300$ and $b = 1200$ and we measure the total number of blocks sent by the nodes in each phase throughout the distribution time. The result is shown in Figure 3. We observe that for both Random and Rarest First, in most phases during the distribution time, close to 300 blocks are sent in each phase. It shows that the uploading bandwidth among the nodes is very well utilized. We believe that Random and Rarest First both succeed in keeping the block content of the nodes heterogeneous, i.e., each node has content not similar to that of their neighbors.

Thus, each node can upload some block useful to its neighbors at each phase, and maintain the high utilization of bandwidth.

For the practical concern, Random can be implemented more easily than Rarest First, while maintaining similar performance. We believe that Random is a better choice than Rarest First in practice.

Sequential. Sequential has much worse distribution time than Random and Rarest First. We observe for the case of $n = 300$ and $b = 1200$, the number of blocks sent per phase is between 50 to 150 for most phases during the distribution time. The low usage of bandwidth is due to the fact that many nodes have similar content as their neighbors. Thus, no useful blocks can be uploaded to their neighbors.

3.2 Routing Heuristics for Push Protocols

Models with push protocols have the advantage that no overhead is needed for sending requests. In this subsection, we study the performance of the heuristics Random, Rarest First and Sequential for push protocols.

We first state clearly how the three heuristics are defined for push protocols. At each phase, each node u first randomly selects a neighbor v. Let S be the set of blocks that u has but v does not. Then, the three heuristics perform as follows.

1. **Random.** Send a random block in S to v.
2. **Rarest First.** Send the rarest block in S to v, where availability is measured according to u.
3. **Sequential.** Send the block in S corresponding to the earilest part of the file to v.

We study the performance of the three heuristics in the model UG-5-Push, with different combinations of n and b. The results are shown in Table 3.

We observe that Random and Rarest First have very similar performance in Push Protocols. In fact, in three out of the four cases tested, Random has smaller distribution time than Rarest first. The reason is that with Push protocols, there are send collisions where the same block is sent to a node from multiple neighbors in a phase. Rarest First makes send collisions more common, thus leading to a lost of efficiency. Sequential has even worse performance because send collision becomes very serious in this case.

4 Concluding Remarks

In this paper we extended [1] on modelling the distribution of data blocks in BT systems. We have studied new network features that BT systems are using and different local routing heuristics. Our simulation confirms that $\Omega(b \log n)$ phases are needed for the model in [1] while showing that the random and rarest block first heuristics under more realistic network setting lead to $b + O(\log n)$ total distribution time. An interesting open problem is to provide a mathematical analysis of these heuristics. Other interesting problems include modelling vary client bandwidth, and dynamic issues like new nodes joining, and nodes leaving the system.

References

1. D. Arthur and R. Panigrahy. Analyzing BitTorrent and Related Peer-to-Peer Networks. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2006, pages 961–969.
2. A. Bharambe, C. Herley, and V. Padmanabhan. Analyzing and Improving BitTorrent Performance. Technical Report MSR-TR-2005-03, Microsoft Research, 2005.
3. BitTorrent. http://www.bittorrent.com.
4. BitTorrent Accounts for 35% of Traffic. http://yro.slashdot.org/yro/04/11/04/1749257.shtml.
5. BitTorrent Protocol Specification v1.0. http://wiki.theory.org/BitTorrentSpecification.
6. C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. Technical Report MSR-TR-2004-80, Microsoft Research, 2004.
7. B. Cohen. Incentives Building Robustness in BitTorrent. http://www.bittorrent.org/bittorrentecon.pdf.
8. M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Hamra and L. Garces-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *Proceedings of Passive and Active Measurements*, 2004, pages 1–11.
9. J. Pouwelse, P. Garbacki, D. Epema and H. Sips. The BitTorrent P2P File-Sharing System: Measurements and Analysis, 2005, 205–126.
10. D. Qiu and R. Srikant. Modelling and Performance Analysis of BitTorrent-like Peer-top-Peer Networks. In *Proceedings of SIGCOMM*, 2004, pages 367–378.
11. X. Yang and G. de Vecianna. Service Capacity of Peer to Peer Networks. In *Proceedings of INFOCOM*, 2004, pages 2242–2252.