

Modeling an Agent-Based Decentralized File Sharing Network

Alex Gonopolskiy Benjamin Nash

December 18, 2007

Abstract

In this paper we propose a distributed file sharing network model. We take inspiration from the BitTorrent model, but use an even more decentralized approach. We implement the model using the NetLogo network modeling package[1]. Our agent-based NetLogo model allows two different types of agents to enter the network: altruistic agents and selfish agents. Selfish nodes refuse to share any of their own data with other nodes, mimicking behavior found in real BitTorrent and networks. We demonstrate that our model is very resilient to large numbers of selfish agents entering the network by showing that the slowdown in information diffusion is small and the probability of the network disconnecting is low.

1 Introduction

File sharing with BitTorrent and other clients has gained popularity in recent years as the protocol has matured and users have become familiar with its purpose. Popular (legitimate) uses of decentralized file sharing networks include the circulation of free multimedia and open-source software. The success of such networks depends on users to upload parts of their partially-completed downloads to other users. However, file sharing protocols can not require this behavior from users, and users may decide not to expend the extra bandwidth that would be required. Therefore, it is important that file sharing networks operate effectively even with a large number of users which are unwilling to share data.

2 Background

Our model depends heavily on ideas from two areas: the BitTorrent file sharing protocol and strategic network formation. Although we do not take the BitTorrent model verbatim or apply the standard techniques of strategic network formation analysis, our model relays heavily on both of these ideas.

2.1 BitTorrent

The BitTorrent file sharing protocol differs from traditional file sharing networks in both the goals of agents and in the way information is disseminated. While traditional file

sharing networks contain agents who are searching for completely separate file, a BitTorrent network contains agents who are all concerned with acquiring the same large file or group of files. The files in a BitTorrent network are also divided into smaller downloadable units called "chunks". Therefore, an agent does not search for another agent who owns the file that it desires; instead, it searches for another agent who owns a chunk that it requires to finish the larger download. A consequence of this aspect is that agents are able to upload to other agents even before completing the entire download.[2]

2.2 Strategic Network Formation

Strategic network formation refers to the idea that agents make autonomous decisions about which other agents to connect to. Usually there are certain criteria that agents are trying to optimize. This could include a global goal such as the distribution of a resource across the network, or a local goal, such as the accumulation of a resource at the local agent at the expense of other agents. Under such models there is usually a cost for each connection. Thus agents try to reap the benefits that come from forming valuable connections without incurring too large of a penalty coming from too many connections. Depending on the cost C associated with each of these connections, the network can either become completely connected, if the cost C is low, or completely disconnected when the cost C outweighs the benefit of making any connections at all.[3] There are a number of different techniques for analyzing these networks, such as Pareto Efficiency, which seeks a stable configuration of the network where no agent stands to gain from adding or losing connections. However, most of the techniques focus on static agents, agents whose internal configuration does not change. Our model is fundamentally different from traditional network formation models because the internal states of agents change after each time step; a network that was stable on one iteration may not be stable on the next simply because agents will exchange files and will now have to make new decisions.

3 Our Model

With our model we addressed the following question: What policies can be added to a file sharing network that will make it more resilient to selfish users without requiring access to centralized information?

With BitTorrent, the location of certain data is found using a distributed hash table, so an agent can gain knowledge about the every other agent. Our model is even model is less centralized, and does not assume that access to such a hash table is available. Instead, agents can only request information of their neighbors such as the data chunks that it possesses or the nodes that it is connected to. Upon entering the network an agent can connect to up to M other random agents. Usually M is set to 1, to make the model completely decentralized, however setting the M to even 2 can have great impact on the model which will be discussed in our results. The network contains one agent called the "seed", which has all the pieces of the file that is being shared. All other agents enter the network with no pieces of the file. Each agent has an upload and download bandwidth, which are currently globally defined for all altruistic nodes. This bandwidth helps determine the speed at which it will allow other agents to download pieces of a file from it.

After entering the network each agent goes through three phases: First, each agent must make decisions about adding new edges or deleting old ones. Second, the edge weights are determined, defining the bandwidth of a connection. Third, each agent attempts to transfer data through each of its connections.

Once a node is done downloading all the pieces it requires, it drops all incoming links, since it no longer has a need to download. We set a probability $p(\text{leave})$ that any finished node will leave the network at a given tick. Similarly, there is a probability $p(\text{enter})$ of a new empty node will entering the network and randomly connecting to M other nodes. This allows us to model a more realistic file sharing environment, where nodes often enter and leave the network.

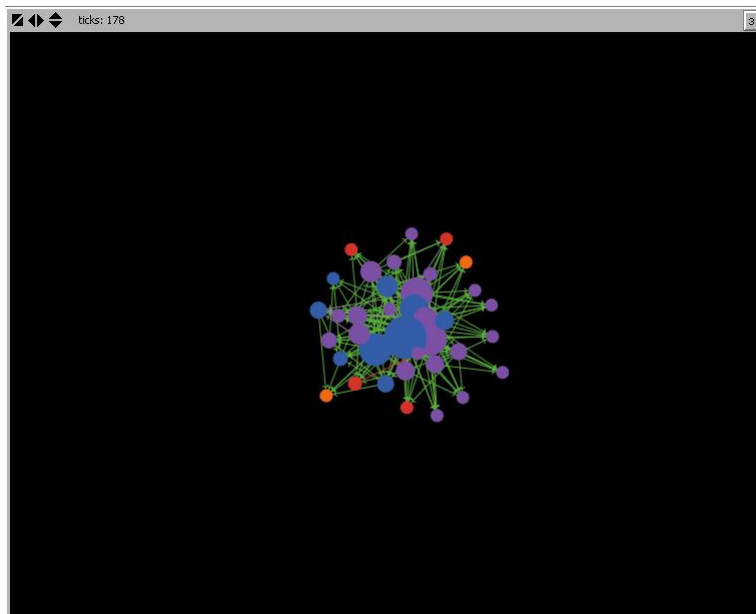


Figure 1: An example run of our model. Red nodes represent selfish nodes. Violet nodes represent altruistic nodes. As nodes download more and more of the file their color gravitates towards green. When they have completed the download they turn different shades of green depending on if it was a selfish node or an altruistic node. The size of the node represents the number of uploading links it has during that tick. Green edges have degree greater than zero, red edges have degree of zero.

3.1 Strategic Network Formation in our Model

Our network is directed, with arcs pointing in the direction of data flow. Therefore, in-links represent edges an agent's downloads, and out-links represent its uploads.

3.1.1 Creating/Deleting Edges

Agents only make decisions about adding or deleting in-links, or connections that they are downloading from. There are three ways that agents modify their incoming links: First, if a neighbor does not have any data that an agent needs, a new connection is formed to a random one of the neighbor's neighbors. The old connection is then dropped. Second,

if the weight of a connection becomes zero, meaning there is no data flowing through that connection, the connection is dropped. The exception to this is if the connection with zero edge weight is a node's last link. In this case, the edge is kept so the agent will not become disconnected from the network. Third, when an agent is not utilizing all of its download bandwidth, it will form a new connection to a random neighbor of a neighbor. Each in-link, however, has an overhead cost C . Thus an agent must balance data flow with connection cost when forming connections.

3.1.2 Adjusting Edge Weights

Edge weights represent bandwidth in our model. Thus, an edge weight should depend on the source node's upload bandwidth and the destination node's download bandwidth, as well as the connection cost C . Calculating bandwidths of connections is important for producing a realistic model. These calculations are done in two steps: First, we make the simplifying assumption that an agent will distribute its upload bandwidth evenly among all of its out-links. So each agent first sets an upper limit on each of its out-links. Second, each node distributes as much of its download bandwidth as it can among its already limited in-links. If the total amount of bandwidth that an agent was allotted through its in-links exceeds its download bandwidth, then it will be able to use all of its download bandwidth and may drop an unused in-link. Otherwise, the agent will have extra download bandwidth and may form a new link, depending on the connection cost.

This is somewhat a simplified model for calculating the edge weights. Notice that after the initial allocation of upload bandwidth, links are not readjusted if the other node does not dedicate any download bandwidth. The main reason we choose to do it in this fashion is for efficiency reasons. It would be take too long to wait for the network to find a stable configuration. A edge weight could also oscillate as each end allocates a different amount of bandwidth based on the network structure.

3.2 Information Diffusion in our Model

After calculating all the edge weights, each agent attempts to download a piece of the file from its neighbors. Since each edge's weight represents the bandwidth allocated to that edge, we view the bandwidth as a probability $0 \leq w \leq 1$ of downloading the needed file chunk on the current simulation tick, where w is the weight of the current edge. This corresponds to a Bernoulli random variable with probability w of achieving success. This allows us to model the time it takes to transfer a file chunk using a geometric distribution, where the expected time between each successful trail is $\frac{1}{w}$. So if the weight of an edge is $\frac{1}{10}$, one would expect 10 ticks to pass before the file chunk is transferred across the link.

3.3 Modeling Selfishness

Modeling selfishness is done through by the upload bandwidth of a node. A selfish node has an upload bandwidth of zero and will not share anything with any other node. It will still allow other nodes to connect to it and retrieve information about its neighbors.

4 Empirical Results

We implemented our model in NetLogo and ran a number of experiments using its built-in plotting and visualization capabilities. Unless stated otherwise, experiments started with the seeder and 30 nodes randomly attached. The download bandwidth of nodes was set to 0.6 and the upload bandwidth was set to 0.3. The connection cost was 0.1. The file was composed of 100 chunks.

4.1 Degree Distribution

The degree distribution of our model is as what one would expect. There are a few nodes that become information hubs as they gain early access to the seeding node, and slowly all other agents aggregate towards them. However, as their degree becomes too large, other agents try to find other nodes to download from as the probability of getting a piece at any given tick becomes rather low. The more interesting behavior happens when the global upload limit is set to something very low, .01 for example. In a scenario like this, if an agent successfully gets a piece of a file from the seeder, you can quickly see it distribute the file piece through the network as everyone is trying to connect to it. Once it shares that file with everyone else, its degree drops back down to normal. Thus you see a pattern of certain information hubs rising and falling depending on which piece they were able to obtain from the central seeding node.

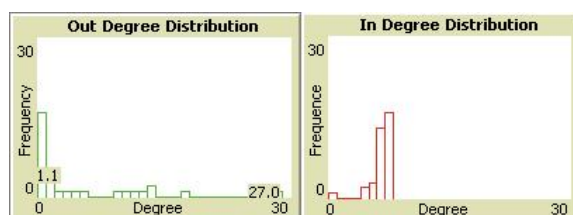


Figure 2: A sample degree distribution output. Very few nodes have a very high out-degree, while the majority of nodes have a low out-degree. This points to the fact that there are information hubs spreading the files through the network.

4.2 Average File Distribution

The average file distribution is measured simply by seeing how many pieces each agent needs against how many pieces there should be total if every node had completed the download. This gives us an estimate of how fast the file is spreading through the network. As one would expect, this is highly dependent on the global upload and download bandwidths. Setting both to high can cause a continual increase in the speed of the file distribution, while setting both to low can slow the network down to a crawl. The more interesting results come when selfish nodes are added to the network. Surprisingly, the network is rather resilient to the addition of selfish nodes. The speed at which the average distribution increases stays relatively high until about seventy five to eighty percent of all agents in the networks are selfish, at which point it begins to slow down dramatically.

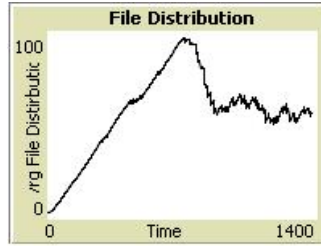


Figure 3: A sample of the file distribution over time. The large dips in the graph represent when a node has completed the file download and has left the network the smaller dips is when a new node enters the network. As you can see the file distribution grows very fast.

4.3 Connected Components

One of the biggest problems we ran into with our model was trying to keep it connected. Since nodes often drop and add new links, there is a chance that the network will separate into more than one connected component. Also, nodes have a probability of leaving the network after they have finished downloading, increasing the potential for network separation. If a certain connected component does not contain a seeder node and every file chunk is not present in at least one of the nodes, the nodes in that connected component have no way of retrieving the entire file. This is an undesirable phenomenon, and we found two additions to model to help avoid this type of disconnection. First, increasing the initial number of in-links M from 1 to 2 significantly reduces the chance of disconnecting. Second, we choose not to drop all downloads with zero bandwidth. If we choose to leave even one such connection, the network becomes noticeably more stable.

4.3.1 Effects of Sorting on Components

One of the more unexpected results that we encountered while working on the model is the dramatic effect of the order in which one allocates bandwidth. Since nodes allocate bandwidth sequentially, the order of the sequence seems to affect whether or not the agents will stay connected to the network or will slowly become disconnected from it. There are a number of possible ways of allocating the download bandwidth, firstly one can just allocate bandwidth in a random order. This seems to have a negative effect overall, since some bandwidth is allocated to very low weighed edges, and we do not have any strategy of which one of the connections that have previously proven to be beneficial to keep. The second possible strategy is one where an agent can simply go in order of the oldest node first. This has the effect of giving the nodes that have the highest probability of having more of the file pieces priority. Since nodes that have been around longer have had more chances at acquiring pieces of the file, prioritizing for those nodes gives a higher probability of having access to those files. Since one of the reasons an agent will drop an edge is based on if its in-link neighbor has a piece of the file we are missing, this strategy greatly increases the stability of the network, and the probability that the nodes will stay connected in one giant component. This strategy does have a trade off, however. By selecting the oldest nodes first, there is a high probability that these nodes have been selected by other agents as well, since they have been around longer and thus have had

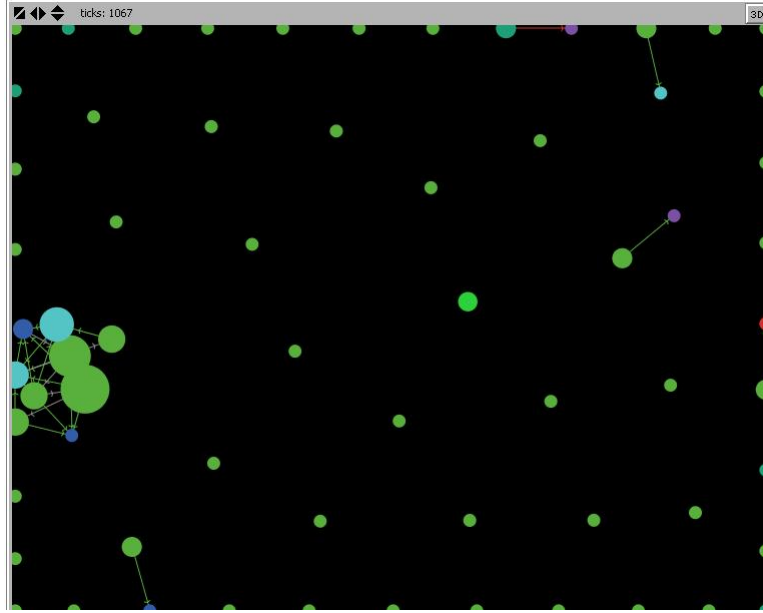


Figure 4: After a certain period of time, some nodes become disconnected from the large connected component and will not be able to get access to all the pieces they require. Furthermore some of them are only connected to selfish nodes which will not share anything.

the chance to acquire more pieces and more connections from incoming new nodes. This means that there is a high probability that the older nodes will have a very low upload bandwidth that they can dedicate to each edge since the upload bandwidth of each edge is dictated by $\frac{Upload_i}{D_{out}(i)}$. This brings us to the last possible strategy. One can allocate bandwidth based on the highest upload bandwidth provided by the neighboring nodes. This is probably the more realistic approach, since it has the benefit of increasing the speed at which the file will spread through the network. Since higher bandwidth means higher chance of obtaining a piece of the file, it also allows our model to be much more tolerant to selfish nodes which was one of the design goals we wanted to achieve. The downside is that it decreases the probability that a certain node will have a piece of the file we want, and thus decreases the overall stability of the network, making the agents change links and move through the network much more often.

4.3.2 Effects of Incoming Nodes on Components

Due to the fact that after an agent is done downloading all of the pieces it requires, it drops all of its links and has a small probability of leaving the network. It is still possible for a new incoming node to become disconnected from the rest of the network by connecting to an isolating seeding node at the start, and then have the seeding node leave the network before it has finished acquiring all the necessary pieces. Once the node becomes disconnected from the large component, our model is structured in such a way that it on its own cannot connect back since it does not have access to any information about other agents, and all information about other agents can only be acquiring through navigating the networks one neighbor at a time. Our solution to this problem was to

increase the number of edges that an incoming node creates from the onset. Increasing the number of edges of an incoming node from 1 to 2 creates a small probability of $\frac{1}{n-1}$ of an incoming node connecting to an isolated node and to the rest of the network. However, considering this event occurs infinitely often, and our assumption is that nodes will not leave the network until they have completed the download, this is enough to bring isolated components back into the network. The drawback of this model is that one can argue it is no longer as decentralized as before, one has to know at two random nodes in the network.

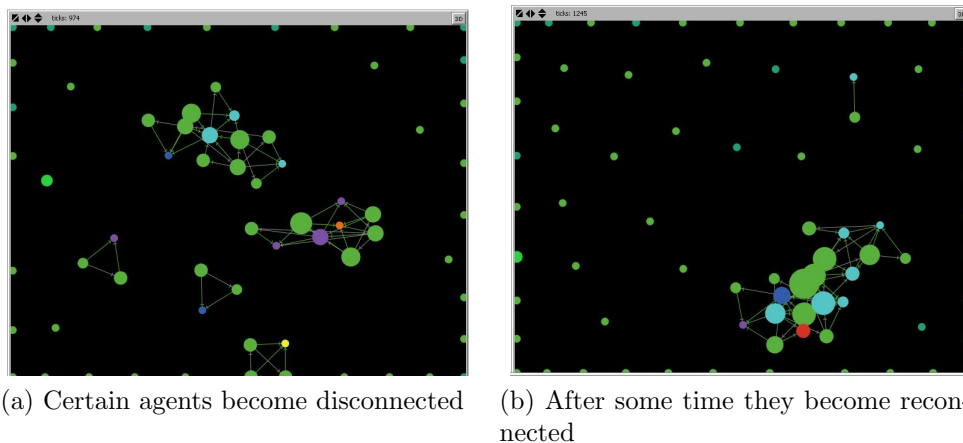


Figure 5: The figure on the right shows that some agents become disconnected, however after a certain period of time as new nodes enter the network they become reconnected to the large component.

5 Conclusion

We have designed our model with the BitTorrent file sharing network in mind. However our model is more decentralized in that it does not keep a hash table allowing one to know the entire structure of the network. We achieve this by introducing a certain level of autonomy to our agents based on the models of strategic network formation previously studied. Furthermore, we investigated our model under a mixture of different agents, those seeking to increase the overall file distribution on the network and those that are selfish and are not willing to share any files. We found that our model is resilient to the introduction of selfish nodes and that the file distribution is relatively unhindered by the introduction of selfish nodes up to a certain phase transition point where the number of selfish agents becomes overwhelming. Furthermore we investigated the structure of the model, mainly its tendency to become disconnected under certain conditions and found ways of limiting this effect or eliminating it all together by introducing an infinitely often occurring event that will allow disconnected nodes to enter back into the network while sacrificing very little of the decentralizes style of the network.

References

- [1] U. Wilensky, “NetLogo,” Evanston, IL., 1999.
- [2] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, “The Bittorrent P2P file-sharing system: Measurements and analysis,” in *4th Int’l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005. [Online]. Available: citeseer.ist.psu.edu/pouwelse04measurement.html
- [3] M. Jackson, “A survey of models of network formation: Stability and efficiency,” in *Group Formation in Economics: Networks, Clubs, and Coalitions*, G. Demange and M. Wooders, Eds. Cambridge University Press: Cambridge, 2003. [Online]. Available: citeseer.ist.psu.edu/article/jackson03survey.html