Avoiding Greediness in Cooperative Peer-to-Peer Networks

Matthias R. Brust[1], Carlos H.C. Ribeiro[1], and Jaruwan Mesit[2]

[1] Technological Institute of Aeronautics
Computer Science Division
50, Praça Mal. Eduardo Gomes, 12228-900 Sao Jose dos Campos, Brazil
{matthias.brust,carlos.ribeiro}@ita.br
[2] University of Central Florida
4000 Central Florida Blvd. Orlando, Florida, 32816, USA
jmesit@cs.ucf.edu

Abstract. In peer-to-peer networks, peers simultaneously play the role of client and server. Since the introduction of the first file-sharing protocols, peer-to-peer networking currently causes more than 35% of all internet network traffic—with an ever increasing tendency. A common file-sharing protocol that occupies most of the peer-to-peer traffic is the BitTorrent protocol. Although based on cooperative principles, in practice it is doomed to fail if peers behave greedily. In this work-in-progress paper, we model the protocol by introducing the game named *Tit-for-Tat Network Termination* (T4TNT) that gives an interesting access to the greediness problem of the BitTorrent protocol. Simulations conducted under this model indicate that greediness can be reduced by solely manipulating the underlying peer-to-peer topology.

Keywords: Peer-to-Peer Networks, Topology Control, Cooperation.

1 Introduction

Peer-to-peer networking where the peers simultaneously play the role of client and server becomes more and more popular. In a peer-to-peer network the participants are peers of equal rights that offer and down content.

Since the introduction of the first file-sharing protocols peer-to-peer networking causes more than currently 35% of the overall network traffic with increasing tendency [1]. However, peer-to-peer approaches are not solely used for file-sharing, but also for VoIP, distributed data storage, video streaming, distributed collaboration, and etc. [2, 3].

A remarkable file-sharing protocol using most of the peer-to-peer traffic is the BitTorrent protocol even used by universities to distribute their lecture material [4]. For a file to download in the peer-to-peer mode, one needs the correspondent pointer file called .torrent file. This file contains information of the tracker that is aware of peers sharing the same file. The tracker directs the client on the host machine to this peer and, generally speaking, manages the file-transfer process on a meta-level. In the BitTorrent protocol, the peer offering the original file ("seed") is called *seeder* and the peer downloading a file is referred as *leecher*. The set of seeders and leechers of one common file are called the swarm of that file. In contrary to many other peer-to-peer file-sharing protocols, the BitTorrent protocol protrudes in the fact to make incomplete downloads available for the swarm to download chunks of the file. This turns the file-sharing process very efficient—in particular for extremely large files—since peers do not have to wait for other peers to finish, before starting to download.

BitTorrent is—as usually any peer-to-peer network—fundamentally based on cooperative behavior and it assumes that leechers turn to be seeders for a file after they have downloaded the file. Although both leechers and seeders are sharing chunks available on their host, seeders still remain most important in the BitTorrent networks because seeders have 100% (all chunks) of a file.

The worst case scenario for the BitTorrent protocol is that leechers leave the network immediately after downloading a file; they simple stop sharing this file. This causes that the remaining peers cannot find missing chunks anymore within the swarm and fragments of incomplete files are left back. This selfish behavior results in successive starvation of the peers, and the swarm dies out. Although based on cooperative principles, in practice the protocol is, thus, failed to doom if peers become greedy. This problem is also known as "the leecher problem".

In order to reduce selfish or greedy behavior, additional mechanisms on the information-flow (network) layer have been suggested and implemented. One strategy, for example, uses the tit-for-tat algorithm that forces cooperation. Hereby, a ratio between the downloaded and uploaded amount of data is calculated and, depending if the ratio is close to one or not, the peer gets preferences by the tracker for downloading with e.g. higher bandwidth.

Summarizing, even considering that the peers maintain a ratio close to one, i.e. the amount of data that has been downloaded is equal to the amount of data that has been uploaded, peers of the swarm can suffer from the problem of starvation and, hence, the dying out of the whole swarm.

The contribution of this work-in-progress paper is to describe a basic mechanism, named T4TNT game (*Tit-for-Tat Network Termination*) which is used to understand the problem of starvation of the swarm, and finally the leecher problem. Based on results from the T4TNT game, we conclude that a crucial impact on the reduction of greedy behavior can be realized by not only providing mechanisms on the information-flow layer, but also on the topology control layer.

Although strongly motivated by the leecher problem caused by the BitTorrent protocol, results of this and future work are not only limited to this field. We see application areas of our research in game theory, social networks, complex networks, and cryptology.

The following section describes the T4TNT game which exhibits some interesting research questions that can partially be reduced to the leecher problem. Section 3 describes the study on the analytical as well as empirical nature of the T4TNT game and relates these to the BitTorrent. Finally, the last section discusses the results and the future work.

2 The T4TNT Game

Given is an undirected network $Net = (N, L)$ with a set of nodes N of size n. The set of links L connects each node with each other, resulting in a clique (later on we change this assumption). Nodes are also referred as participants. We assume that each participant owns a unique chunk item, i.e. nobody else has this item initially. The overall objective of each node is to collect all chunk items which are available in the network to complete e.g. a file or task.

The exchanging of chunks is only permitted in a peer-to-peer mode if both participants are able to exchange interesting chunks to the other. The strategy applied is the tit-for-tat principle where nodes are allowed to exchange also received chunks.

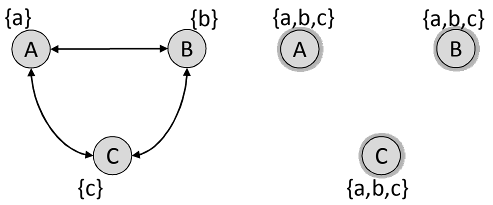
As an additional constraint, if a node terminates, it will immediately leave the network and is not able to exchange any chunk anymore (greedy behavior). This configuration is called the T4TNT game (for *Tit-for-Tat Network Termination*).

2.1 Problem Definition

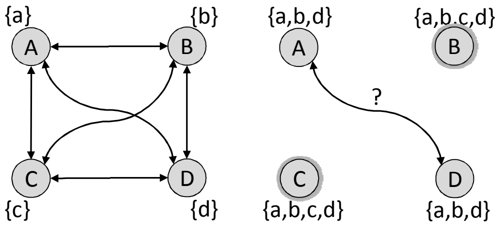
The constructed scenario creates interesting questions and problem situations. As shown in Fig. 1, consider the case where the number of nodes is n = 3. These are some ways of how the nodes can exchange the chunks, but in the end all nodes necessarily reach the terminal state, i.e. each node has all the information available in this network.

The case with n = 4 reveals a more complex nature of the problem (cf. Fig. 1). Not all combinations of steps lead stringently to a final state with all nodes have got all information.

The table in Fig. 1 describes a sequence of steps for the network configuration with n = 4 where finally node B and node C terminate while node A and D do not reach the



A	B	C
a	b	c
ab	**ab**	c
ab	**abc**	**ca**
abc	--	**cab**



A	B	C	D
a	b	c	d
ab	**ab**	c	d
ab	**abc**	**ca**	d
ab	abc	**cad**	**da**
ab	**abcd**	**cadb**	da
abd	--	--	**dab**

Fig. 1. Network with 3 nodes playing the T4TNT game (above) always terminates. A network with 4 nodes reveals situations where not all nodes are able to terminate.

terminal state. However, the situation is starved because of the remaining nodes A and D cannot exchange additional chunks anymore based on the constraints given before. Thus, the remaining swarm dies out. The questions appearing in context of this example are: How many permutations of all steps lead to a final state where all nodes terminate (for n = 4, 5, 6, …)? It seems that the final state depends on the topology. Are there topologies that support the concurrent termination of all nodes more effectively than other topologies? How to construct such topologies?

In particular, the last question introduces the area of topology control (TC) in peer-to-peer networking. Topology control is the science of manipulating a topology in order to optimize it according to a certain [5].

2.2 Reducing the T4TNT Game to the Leecher Problem

The mapping from the T4TNT to the BitTorrent peer-to-peer networking is done when interpreting that the network in T4TNT is the swarm in the BitTorrent network. However, there are some differences that do not enter our model (the T4TNT game) so far.

As mentioned in the introduction, the BitTorrent protocol tries to tackle the leecher problem by introducing additional constraints on the network layer. The T4TNT only allows changes on the underlying topology.

The BitTorrent protocol gives the client the option to choose the order of how to download the missing chunks. A default policy is to use random order where as an improved policy is to download pieces in rarest first order. Usually, the client is able to determine this. The described version of T4TNT does not use additional information about the exchanging order of chunks.

Additionally, there are some very specific issues in the BitTorrent protocol. For example, when a download is almost complete, there is a tendency for the last few chunks to trickle in slowly. To speed this up, the client can send requests for all of its missing chunks to all of its peers [6]. The T4TNT game does not use this or similar kind of “end games”.

3 Study

In the following study we analyze the behavior of the T4TNT game on different network topologies in particular clique graphs, geometric random graphs and random networks (see Fig. 2).

In the first experiment we run the T4TNT game on a clique graph for different number of nodes as shown in Table 1. The measure *Terminations* means the percent of nodes that reach the termination state, i.e. collecting all chunks. The measure *Exchanges* informs of how many tit-for-tats, i.e. transitions, of all possible transitions have to be completed (in a clique network with n nodes there are exactly $n(n-1)/2$ if all nodes reach the termination state). Both measures are in percentage. All simulations have to done with 10000 simulation results. The data given in Tables 1, 2, and 3 are the average values.

Table 1 shows the probability that all nodes terminate in a game with four nodes is about 81%. In other words, about 81% of all possible transition sequences terminate.

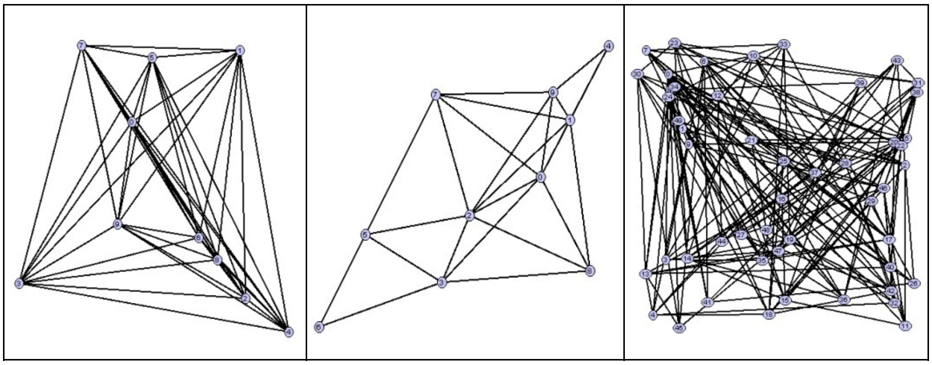


Fig. 2. Clique, geometric random graph and random network

Table 1. Clique networks

	n=4	n=5	n=6	n=10	n=15	n=25
Terminations [%]	81.3	54.6	35.5	16.1	11.9	11.9
Exchanges [%]	97.7	95.9	95.1	95.9	97.5	99.0

(In case of n = 4, this are exactly 720 sequences that terminate.) Interestingly, this number is decreasing by increasing the number of nodes while the total number of exchanges is always more than 95%.

In the second experiment we run the T4TNT game on a clique graph with n = 10 and a geometric random graph for n = 10 (see Table 2). The geometric random graph model is often used in order to model wireless ad hoc or sensor networks [7, 8]. The results show that a typical wireless ad hoc network builds an inappropriate topology for peer-to-peer networking, compared to the clique network.

Table 2. A clique network compared with a geometric random graph

	Clique n=10	GRG n=10
Terminations [%]	16.1	0.1
Exchanges [%]	95.9	63.1

In a next step, we try to estimate if the resources of the network—in particular the links—are used in an efficient manner or if it is possible to obtain similar results by using fewer resources. For this purpose, we introduce the clustering coefficient as local measurement of the clustering of neighbors. The formal definition is given below.

Definition (Local Clustering Coefficient). The *local clustering coefficient* CC of one node v with k_v neighbors is

$$CC_v = \frac{|E(\Gamma_v)|}{\binom{k_v}{2}} \tag{1}$$

where $|E(\Gamma_v)|$ is the number of links in the neighborhood of v and $\binom{k_v}{2}$ is the number of all possible links.

The clustering coefficient CC then is the average local clustering coefficient for all nodes [9]. Many classes of regular networks e.g. square grid networks or the restricted class of circular networks studied in [10] have a high clustering coefficient, i.e. nodes have many mutual neighbors. The opposite extreme to regular networks are random networks, which already have a small clustering coefficient.

However, as we show, the clustering coefficient of a random network can even be more reduced. This corresponds to the fact that randomly appearing groups or cluster have to be deliberately destroyed.

For this purpose, the declustering algorithm "*DeClust*" has been designed and implemented.

DeClust provides a generic approach to implement this general idea of minimizing the clustering coefficient. The basic idea of the algorithm is to verify if a link (u, v) is efficient in terms of the clustering coefficient. In that case, removing the link (u, v) is taken into consideration (lines 3-8). It is not removed immediately because removing might be advantageous for that particular node only. However, since a link removing affects the local clustering coefficients of the 2-hop neighborhood of the set $\{u, v\}$ an additional remove-confirmation phase will be performed (line 11).

In the confirmation phase, nodes exchange the remove-candidate with that corresponding neighbor. For example, if node v calculates that link (v, u) is a remove-candidate, it sends this result to node u. In case that node u has calculated the link (v, u) as remove-candidate as well, the final phase is executed; otherwise the link (v, u) is not removed.

This optimization is justified by the fact that removing a 1-hop link causes a considerably higher impact on the local CC-value than removing a 2-hop link [11]. Since a link removal affects the CC-values of the 2-hop neighborhood of nodes $\{u, v\}$, the final phase calculates the 2-hop neighborhood of $\{u, v\}$. Both the current CC-values of the 2-hop neighborhood as well as the value after a hypothetical removal of (v, u) are calculated. Based on a comparison of those values, it is finally decided whether to remove the link (v, u) (lines 12-18).

As additional criterion, connectivity is guaranteed by line 16 in the pseudo code where removing (u, v) requires that at least one neighbor of u is connected to one neighbor of v.

Furthermore, note that 2-hop synchronization is required before finally removing a link (cf. line 17-18), since 2-hop topological information is required in order to plan the action. Then again, this local link removal affects the 2-hop neighbors. The synchronization procedure is not detailed here.

```
Algorithm. DeClust (for node $v$)

Input:    $N$: Initial set of neighbor nodes
          $N2$: Neighbors of $u \in N$ (to calc $CC$)
          $RC$: Set of remove candidates
Output:   $N_{DeClust}$: Resulting neighbor nodes

01:  $CC(v) \longleftarrow |E(\Gamma_v)|/\binom{k_v}{2}$
02:  $RC(v) \longleftarrow \emptyset$
03:  for each $u \in N$ do
04:      $N(v) \longleftarrow N(v) \setminus \{u\}$
05:      $CC_{rc}(v) \longleftarrow |E(\Gamma_v)|/\binom{k_v}{2}$
06:      if $CC_{rc}(v) < CC(v)$ then
07:          $RC(v) \longleftarrow RC(v) \cup \{u\}$
08:      $N(v) \longleftarrow N(v) \cup \{u\}$
09:  $sendRC\big(v, RC(v)\big)$
10:  $receiveRC\big(u, RC(u)\big)$
11:  for each $w \in \big(RC(v) \cap RC(u)\big)$ do
12:      $CC(uv) \longleftarrow \sum_{x \in (N(v) \cup N(u))} CC(x)$
13:      $N(v) \longleftarrow N(v) \setminus \{w\}$
14:      $CC_{rc}(uv) \longleftarrow \sum_{x \in (N(v) \cup N(u))} CC(x)$
15:      if $CC_{rc}(uv) < CC(uv)$ then
16:          if $N(v) \cap N(u) = \emptyset$ then
17:              if $permit(v) =$ true then
18:                  $N(v) \longleftarrow N(v) \cup \{w\}$
19:  $N_{DeClust}(u) \longleftarrow N(v)$
```

Applying *DeClust* to a random network given by the number of nodes *n* and the average node degree *k* results in the following data given in Table 3.

Table 3. Network characteristics for topologies before and after applying *DeClust*

	n=50
	k=5
		n=100
		k =5
			n=50
			k=10
				n=100
				k=10

Terminations before [%]	59.4	57.0	78.8	78.4
Terminations after [%]	54.2	52.4	77.8	74.8
Exchanges before [%]	90.1	90.2	96.8	97.0
Exchanges after [%]	89.3	89.5	97.0	97.0
CC before	0.47	0.45	0.34	0.18
CC after	0.16	0.15	0.12	0.04
k before	9.0	9.5	16.5	18.1
k after	6.8	7.6	11.4	12.5

Table 3 shows the number of terminations, transitions, clustering coefficient and the average node degree for different network configurations for before and after applying *DeClust*. The general conclusion is that for networks with low *k DeClust* reduces the number of termination of about 5% whereby networks with higher k suffer a reduction of between 1% and 4%. The number of transitions for all networks does not change significantly, what means that the information exchange has not been affected at all by *DeClust*. However, looking at the average node degree *k*, one can see that the *DeClust* networks need up to 30% link resources by having almost the same terminations result as the non-*DeClust* networks.

Saving such an amount of network resources results in the assumption that—if we can find an algorithm that smartly puts about 10% of the saved resources back again in the network—a topology-driven approach can lead to an increase of the number of terminations by using fewer resources.

4 Conclusions and Future Work

This work-in-progress paper introduces the T4TNT game which is used in order to model greedy or selfish behavior in a peer-to-peer network. We show that T4TNT can partially be reduced to the BitTorrent protocol. The BitTorrent protocol is founded upon cooperative concepts. But in practice, cooperation cannot be taken for granted. Thus, this peer-to-peer file sharing protocol suffers efficiency when peers leave the network after finishing their downloads.

We show with empiric studies on T4TNT what the nature of this problem is and propose to use topology control mechanisms in order to reduce the potential greediness factor in the overall system.

For demonstrating, this paper introduces the localized algorithm called *DeClust* which reduces the network resources used while maintaining the efficiency rate of node termination.

The main effect of *DeClust* is to de-cluster the randomized networks regarding to the clustering coefficient. However, *DeClust* can be applied to any other network as well. *DeClust* reduces the clustering coefficient by removing dedicated links, effecting in deliberately destroying groups or local clusters.

DeClust is able to save about 30% of links while maintaining the node termination rate. Saving such an amount of network resources results in the assumption that—if we can find an algorithm that smartly puts about 10% of the saved resources back again in the network—a topology-driven approach can lead to an increasing number of terminations by using considerably fewer resources.

In future works, we plan to extend *DeClust* with the ability to set the saved resources in a smart manner such that more nodes are able to terminate. Additionally, we plan to consider different network types, as small-world and scale-free networks. Furthermore, we think on trying to describe the T4TNT game by regular expressions and to consider the complexity of the problem.

References

1. Izal, M., Urvoy-Keller, G., Biersack, E.W., Felber, P.A., Hamra, A.A., Garces-Erice, L.: Dissecting BitTorrent: Five Months in a Torrent's Lifetime. LNCS, pp. 1–11 (2004)
2. Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In: Proceedings of IEEE INFOCOM (2006)

3. Liogkas, N., Nelson, R., Kohler, E., Zhang, L.: Exploiting bittorrent for fun (but not profit). In: Proc. 5th Itl. Workshop on Peer-to-Peer Systems, IPTPS (2006)
4. University, S.: Stanford Engineering Everywhere (2008)
5. Santi, P.: Topology Control in Wireless Ad Hoc and Sensor Networks. ACM Computing Surveys 37, 164–194 (2005)
6. Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analyzing and Improving BitTorrent Performance. Microsoft Research, Microsoft Corporation One Microsoft Way Redmond, WA 98052, 2005-2003 (2005)
7. Santi, P.: Topology Control in Wireless Ad Hoc and Sensor Networks. Wiley, Chichester (2005)
8. Penrose, M.: Random Geometric Graphs. Oxford University Press, Oxford (2003)
9. Watts, D.J.: Small Worlds - The Dynamics of Networks between Order and Randomness. Princeton University Press, Princeton (1999)
10. Watts, D.J., Strogatz, H.: Collective Dynamics of 'Small World' Networks. Nature 393, 440–442 (1998)
11. Calinescu, G.: Computing 2-Hop Neighborhoods in Ad Hoc Wireless Networks. In: Pierre, S., Barbeau, M., Kranakis, E. (eds.) ADHOC-NOW 2003. LNCS, vol. 2865, pp. 175–186. Springer, Heidelberg (2003)