# BitHoc: BitTorrent for Wireless Ad Hoc networks

Mohamed Karim SBAI*†, Chadi BARAKAT*‡, Jaeyoung CHOI*§, Anwar AL HAMRA*¶ and Thierry TURLETTI*‖
*Project-Team Planète, INRIA Sophia Antipolis, France
Email: †Mohamed_Karim.Sbai@sophia.inria.fr, ‡Chadi.Barakat@sophia.inria.fr, §Jaeyoung.Choi@sophia.inria.fr,
¶Anwar.Al_Hamra@sophia.inria.fr, ‖ Thierry.Turletti@sophia.inria.fr

***Abstract*—BitTorrent is one of the Internet's most efficient content distribution protocols. It is known to perform very well over the wired Internet where end-to-end performance is almost guaranteed. However, in wireless ad hoc networks, many constraints appear as the scarcity of resources and their shared nature, which make running BitTorrent in such an environment with its default configuration not lead to best performances. To these constraints it adds the fact that peers are both routers and end-users and that TCP-performance drops seriously with the number of hops. We show in this work that the neighbor selection mechanism in BitTorrent plays an important role in determining the performance of the protocol when deployed over a wireless ad hoc network. It is no longer efficient to choose and treat with peers independently of their location. A first solution is to limit the scope of the neighborhood. In this case, TCP connections are fast but there is no more diversity of pieces in the network: pieces propagate in a unique direction from the seed to distant peers. This prohibits peers from reciprocating data and leads to low sharing ratios and suboptimal utilization of network resources. To recover from these impairments, we propose BitHoc, an enhancement to BitTorrent, which aims to minimize the time to download the content and at the same time to enforce cooperation and fairness among peers. BitHoc considers a restricted neighborhood to reduce routing overhead and to improve throughput, while establishing few connections to remote peers to improve diversity of pieces. To support this, BitHoc modifies the choking algorithm and adds a new piece selection strategy. With the help of extensive NS-2 simulations, we show that these enhancements to BitTorrent significantly improve the file completion time while fully profiting from the incentives implemented in BitTorrent to enforce fair sharing.**

## I. Introduction

Wireless ad hoc networks and P2P file sharing applications are two emerging technologies based on the same paradigm: the P2P paradigm. This paradigm aims to establish large scale distributed services without the need for any infrastructure. Within this paradigm, users have symmetric roles. The global service is ensured thanks to their collaboration. In the case of a wireless ad hoc network, the network is a set of wireless nodes with no central administration or base station. Nodes in such a network operate both as routers and hosts. Multi-hop routing approaches are used to ensure connection between distant nodes. For P2P file sharing applications, peers collaborate in downloading data and multimedia content. Each peer shares some of its upload capacity by serving other peers. The global capacity of the system grows then exponentially with the number of peers[1]. Gnutella [8], Freenet [9] and BitTorrent [1] are examples of P2P content sharing applications in the Internet.

Both P2P file sharing applications and wireless ad hoc networks are mature fields of research. They have been studied heavily but separately in the literature. Only few works try to study how they perform together (e.g., [14] [15] [16]). These works focus on the content lookup problem in wireless ad hoc networks without studying the efficiency of the content sharing itself. Studying the performance of file sharing applications over wireless ad hoc networks is challenging because of the importance of both areas from user and operator perspectives, and the diverse constraints imposed by the use of wireless channels. Indeed, as nodes are both routers and end-users, the routing overhead must be taken into consideration. Furthermore, the performance of transport protocols such as TCP drops seriously when multi-hop paths are used. That is why current topology-unaware P2P file sharing applications are not expected to perform well when deployed over wireless ad hoc networks. Designing efficient file sharing solutions for such networks is an important area of research.

In this work, we investigate how well a P2P file sharing solution developed for the wired Internet performs over a wireless ad hoc network. Our aim is to come up with a solution that minimizes the content download time while at the same time improving collaboration by enforcing fair sharing among peers. As efficient and fair content sharing is targeted, we choose to adapt BitTorrent [1] as a file sharing protocol given its large usage and its known close to optimal performances in the wired Internet [17]. When data is distributed using BitTorrent, interested peers supply pieces of the data to other peers, reducing the burden on any individual peer, providing redundancy in the network, and reducing dependency on the original seed. In addition, BitTorrent implements incentives that encourage peers to collaborate in downloading the content.

This paper considers the particular case when every ad hoc node is interested in downloading the content. One can see it as an extreme case where the load is maximal and the impact of the underlying topology is more pronounced. We aim at well understanding this case and proposing an efficient solution for it before moving into less loaded scenarios in future work. The performance evaluation is done through extensive NS-2 simulations using regular modules for the ad hoc routing and

[0]This work was supported by the ITEA project Expeshare and by INRIA within the PURPURA COLOR Collaboration.

[1]This capacity remains constant with the classical client-server model.

wireless medium and our implementation of BitTorrent in NS-2[2].

Our main contributions in this work can be summarized as follows. BitTorrent with its default configuration is topology unaware. It establishes TCP connections with neighbors independently of their location in the network. This choice of neighbors can lead to slow TCP connections due to long multi-hop paths and routing overhead. Sharing can also be bad when using large pieces since complete pieces can not be sent too far to be reused later by other peers. A first solution is to limit the scope of the neighborhood. In this case, we noticed shorter download times but sharing is bad since there is no diversity of pieces in the network. The pieces of the shared file propagate in a unique direction from the seed to the farthest nodes. To recover from these impairments, we propose BitHoc, an enhanced variant of BitTorrent tuned to ad hoc networks. BitHoc considers a restricted neighborhood to diminish routing overhead and to improve throughput, while establishing few connections to remote peers to improve diversity of pieces. To implement this, BitHoc modifies the choking algorithm and adds a new piece selection strategy. With the help of simulations, we show that these enhancements to BitTorrent considerably improves the file completion time while fully benefiting from the incentives implemented in BitTorrent to enforce fair sharing. In a practical scenario where wired and wireless networks are merged together, BitHoc can run over wireless interfaces connecting to ad hoc networks while regular BitTorrent runs over all other network interfaces.

Section II of this paper presents an overview of the state of the art in deploying P2P solutions over wireless ad hoc networks. Section III describes briefly the BitTorrent protocol. The framework of the study is discussed in Section IV. Section V shows the importance of the piece size in determining the performance of BitTorrent. Section VI studies the impact of the scope of the neighborhood. Section VII presents our solution BitHoc. Section VIII summarizes the work and gives some ideas on our future work.

## II. State of the art

In this section, we present an overview of the state of the art of P2P file sharing applications and their different implementations in wireless ad hoc networks.

*1) P2P applications in the Internet:* There are several design approaches for the construction of P2P overlays over the Internet. One can distinguish between structured and non-structured overlays. This classification is done from the standpoint of resources lookup. In non-structured overlays like Gnutella [8], there is no control on the structure of the overlay. Peers discover each other by flooding the network and by learning from previous sessions. The P2P application in this case is not conscious of the topological location of the other peers. In case of structured overlays, an overlay routing algorithm is introduced to locate the content in the network. Several structured overlay networks have been proposed like CAN [10], Chord [11], Pastry [13] and Tapestry [12]. All of them use Distributed Hash Tables (DHT) in their routing of lookup requests. Such tables allow the lookup to scale logarithmically with the number of nodes in the overlay. Again most of these structured overlays are topology independent. On the other hand, there is BitTorrent [1] that does not concentrate on the information lookup since it uses a centralized tracker to discover neighbors. However, it concentrates on optimal utilization of the network capacity when sharing the file between the different interested peers. Since we are mainly concerned in this work by the data transfer plane, we adopt BitTorrent and we extend it to wireless ad hoc networks. More details on BitTorrent are presented in Section III.

*2) P2P applications in MANET:* MANET stands for mobile ad hoc networks. Both structured and non-structured overlays have been implemented in MANET. Since nodes are both end-users and routers, some cross-layer design approaches have been introduced. These approaches suppose that P2P applications operate both at the network layer and at the application layer. One can divide the design space into four subspaces:

- Non-structured and layered design: Oliviera and al. study in their work [14] the performance of Gnutella deployed over three ad hoc routing protocols DSR [3], AODV [4] and DSDV [2]. Their results show that the ratio of delivered packets is lower than those of unicast applications deployed over MANET. This is due to the fact that Gnutella chooses neighbors independently of their locations. The overlay construction is topology independent.
- Non-structured and cross-layer design: The work done by Klemm and al. in [15] proposes to integrate the peer lookup mechanism of a P2P application like Gnutella in the network layer and compares this design to the layered design proposed by Oliviera and Al. They propose ORION that establishes connections on demand through the routing mechanism. The cross-layer lookup implemented by ORION is shown to provide higher successful transfers ratio than in the layered scenario.
- Structured and layered design: A proximity-conscious DHT (Pastry) has been deployed over the DSR routing protocol in [16]. As it is a layered design, there is no interaction between the DHT and the routing protocol. This leads to an overhead in maintaining routes for both the application layer and the network routing layer.
- Structured and cross-layer design: This design is named Ekta by Das and al in [16]. The functionalities of the Pastery DHT are integrated within the routing protocol. The main idea is the mapping of the peer identifiers in the same namespace than the IP addresses. Their results show that Ekta is better than the layered design in terms of number of successfully delivered packets.

*3) Former studies on BitTorrent over wireless ad hoc networks:* Several works tried to adapt BitTorrent to wireless ad hoc networks. [18] and [19] deploy BitTorrent on the top of such networks. They only focus on the tuning of the

[2]Simulation code and scripts will be available in the public domain.

peer discovery phase without addressing the efficiency of the content sharing itself. Michiardi and Urvoy-Keller in their work [7] study the performance of a cooperative mechanism to distribute content from one source to a potentially large number of destinations. They propose to deploy BitTorrent with a minor change allowing neighbor discovery and traffic locality. This is done by selecting only near neighbors as effective neighbors. The result is a decrease in the total download time and energy consumption. Their work is relevant to ours; however we go one step further by focusing not only on the download time but also on the sharing among peers. Our proposed solution BitHoc performs better in both regards.

## III. BitTorrent: a content distribution protocol

BitTorrent (see e.g., [1], [17]) is a scalable P2P content distribution protocol. Each client shares some of its upload bandwidth with other peers interested in the same content in order to increase the global system capacity. Peers cooperating to download the same content form a *torrent*. A peer discovers other peers by contacting a central rendezvous node called *tracker*. The latter stores IP addresses of all peers in the torrent and maintains statistics on uploads and downloads. To facilitate the replication of the content in the network and to ensure multi-sourcing, a file is subdivided into a set of pieces. Each piece is also subdivided into blocks. A peer which has all pieces of the file is called a *seed*. When the peer is downloading pieces, it is called a *leecher*.

Each peer maintains a peer list. Neighbors are those of this list with whom the peer can open a TCP-connection to exchange data and information. Only four simultaneous outgoing active TCP connections are allowed by the protocol. The corresponding neighbors are called *effective neighbors*. They are selected according to the *choking algorithm* of BitTorrent. This algorithm is executed periodically. Once the *choking period* expires, a peer chooses to unchoke the 3 peers uploading to him at the highest rate. It is a best slot unchoking. This strategy, called *tit-for-tat*, ensures reciprocity and enforces collaboration among peers. Now to discover new upload capacities, a peer chooses randomly a fourth peer to unchoke. This unchoking slot is called optimistic slot. All other neighbors are left choked. When unchoked, a peer selects a piece to download using a specific piece selection strategy. This strategy is called *local rarest first*. Indeed, each peer maintains a list of pieces owned by all its neighbors. When selecting a piece, a peer chooses the piece with the least redundancy in its neighborhood. In case of equality, one of the rarest pieces is chosen randomly. Rarest first is supposed to increase the entropy of pieces in the network. Here are some performance metrics of BitTorrent that we will use in our study:

- $U_{ij}$: Total bytes uploaded by peer i to peer j.
- $D_{ij}$: Total bytes downloaded by peer i from peer j. ($U_{ij} = D_{ji}$)
- $R_{ij}$: Ratio of sharing between peer i and peer j.

$$R_{ij} = \frac{min(U_{ij}, D_{ij})}{max(U_{ij}, D_{ij})} \tag{1}$$

  It is easy to verify that $R_{ij} = R_{ji}$ .
- $N_i$: Number of neighbors j of peer i such that $U_{ij} \neq 0$ or $D_{ij} \neq 0$.
- $R_i$ : Sharing ratio for node i.

$$R_i = \frac{1}{N_i}. \sum_{j|\ U_{ij}\neq 0\ or\ D_{ij}\neq 0} R_{ij} \tag{2}$$

- $F_i$: The finish time of peer i. It is the time by which peer i receives all pieces of the file.

As we are studying BitTorrent over wireless ad hoc networks where topology matters, we consider some additional performance metrics related to topological positions of peers. The file is supposed to exist at one seed S at the beginning of the session. Our metrics quantify the quality of service perceived by peers as a function of their relative positions with respect to the seed.

- $F_h$: Average finish time of peers (or nodes) located at h hops from seed S.

$$F_h = \frac{1}{n_h}. \sum_{i|\ H(i)=h} F_i \tag{3}$$

  where $n_h$ is the number of peers located at h hops from seed S and H(i) a function that gives the number of hops between any node i and seed S.
- $R_h$: Average sharing ratio of peers (or nodes) located at h hops from seed S.

$$R_h = \frac{1}{n_h}. \sum_{i|\ H(i)=h} R_i \tag{4}$$

## IV. Framework of the study

This section is organized as follows. First, we discuss our choice of implicating all ad hoc nodes in downloading the content. Then, we describe preliminary changes we made to BitTorrent to allow peer discovery and the exchange of signaling over wireless ad hoc networks. After that, we discuss the stack of protocols we use in our deployment of BitTorrent. The next part gives an idea about the NS-2 module designed for the study. Finally, we introduce our main scenario for the utilization of BitTorrent over wireless ad hoc networks. The results shown in the following sections are those of simulations run over this scenario.

### A. All nodes are peers

We consider the case study where all nodes are peers interested in the same content. In addition, nodes run ad-hoc routing and relay the packets of each other at the routing layer. In this case study, the underlying topology has a big impact on the performance of BitTorrent. Indeed, any piece sent over a suboptimal route will cause resource consumption in all intermediate nodes. When all nodes are peers, this will affect

all peers located on these nodes by stealing bandwidth from them without being able to profit from this transmission since it happens at the routing layer. However, if intermediate nodes are not peers interested in the same content, this suboptimal piece transmission will have less impact on the torrent itself since it does not directly steal bandwidth from peers. The interaction between peer selection and routing overhead is consequently of less importance. Add to this the fact that when all nodes are peers, the traffic generated by the torrent is maximal and an optimization is more needed.

### *B. Trackerless BitTorrent*

Wireless ad hoc networks are infrastructureless. One can not rely on a centralized tracker when applying BitTorrent to such networks. So, we opt in our study for a trackerless approach. Since the most important role of a tracker in the Internet is to provide peers with the identifiers of other peers, we need to introduce a peer discovery mechanism. In our evaluation framework, to discover new peers, a peer floods periodically the network with a HELLO message. This message is transmitted to wireless neighbors with some initial TTL to control the scope of the flood. Receiving a HELLO message, a peer decrements the TTL and forwards it to its wireless neighbors, and so on. The message is dropped and not forwarded when its TTL reaches zero. After receiving a HELLO message, a peer answers in unicast to its initial source with a HELLO REPLY message containing its identity and the list of pieces in its possession. This list is important for the piece selection strategy. Clearly, the initial TTL value of the HELLO message decides of the scope of the neighborhood to discover.

### *C. Stack of protocols and packets exchanged between peers*

In BitTorrent, peers exchange two types of packets: Data packets and control packets. We choose in our NS-2 implementation to send data packets via TCP connections because reliability and congestion control are needed when transporting blocks of file. However, control packets as for peer discovery and piece updates contain small and urgent information that is better to transport using UDP. Here are the different control packets exchanged between peers:

- **HELLO**: see IV-B.
- **HELLO REPLY**: see IV-B.
- **UPDATE PIECE LIST**: when a peer receives a new piece, it sends an UPDATE PIECE LIST to all its P2P neighbours.
- **PIECE OFFER REQUEST**: when a peer i unchokes a peer j, it sends a PIECE OFFER REQUEST packet to j. This packet contains the list of pieces that i has already downloaded.
- **PIECE OFFER REPLY**: receiving a PIECE OFFER REQUEST, a peer answers with a PIECE OFFER REPLY packet. After applying the piece selection strategy, it decides whether to accept or to reject the offer. A flag included in the PIECE OFFER REPLY packet indicates this decision (ACCEPT or REJECT). In the case the offer is accepted, the peer indicates the number of the requested piece. During the choking period, many PIECE OFFER REPLY packets can be sent the offering peer in order to allow the transmission of several pieces.

### *D. The NS-2 module design*

To run simulations in NS-2, we added a module containing C++ code for different classes allowing BitTorrent to run at the application layer of an ad hoc node. We call BitHoc this variant of BitTorrent that contains all the modifications we propose. All data structures and treatments have been added to simulate the functionalities of BitTorrent. A TCL script is used to initialize the parameters of the protocol. Here are some of the important parameters:

- **max_upload_num_**: The maximum number of active upload connections per peer. It is also the number of parallel choking slots.
- **choking_period_**: The period of the choking algorithm. It is the interval of time during which a peer tries to offer pieces to a set of effective neighbors.
- **choking_best_neighbors_num_**: The number of neighbors chosen as effective neighbors during a choking period because they are best uploaders.
- **received_bytes_reset_interval_**: In the choking algorithm, each peer chooses its best neighbors with whom to reciprocate data based on how many bytes they send to it in some specific interval.
- **flooding_ttl_**: The maximum number of hops that will be used while flooding the network with HELLO messages.
- **piece_num_**: The number of pieces in the file to be shared.
- **block_num_**: The number of blocks in one piece.
- **block_size_**: The size of a block in bytes. The size of the shared file is equal to piece_num_ * block_num_ * block_size_.

### *E. The main scenario*

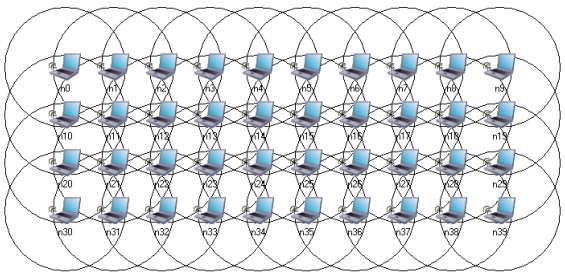

Fig. 1. Topology for simulations

We consider a network of 40 nodes distributed in a plane following a grid as shown in Figure 1. The distance between two physical neighbors is set to 40 m for a range of wireless transmissions equal to 50m. This ensures connectivity while minimizing interference. At the beginning of simulations, node 0 located at the top left is the seed and the other nodes are leechers. The file size is set equal to 10 Mbytes, which can

be seen as a small video clip. All peers start downloading the file at the same time t=1500s by first looking for each other then sharing the pieces of the file according to the BitHoc algorithms. This interval at the beginning gives the network enough time to stabilize by calculating its routing tables. BitTorrent parameters are set as indicated in table I. Concerning the underlying layers, the nodes connect to each other using the 802.11 MAC Layer with the RTS/CTS-Data/ACK mechanism enabled. The data rate is set to 1 Mb/s. For ad hoc routing, we use the DSDV proactive protocol [2] that provides each node with the list of shortest path routes to all other nodes in the network.

TABLE I
BITTORRENT PARAMETERS

| Parameter | Value |
|---|---|
| max_upload_num_ | 4 |
| choking_period_ | 40s |
| choking_best_neighbors_num_ | 3 |
| received_bytes_reset_interval_ | 80s |
| flooding_ttl_ | depends on simulation |
| piece_num_ | depends on simulation |
| block_num_ | depends on simulation |
| block_size_ | 1KB |

## V. IMPACT OF PIECE SIZE

We start by evaluating regular BitTorrent where the overlay is constructed without considering the underlying wireless topology. We give a particular attention to the piece size and to its impact on both the finish time of peers and their sharing ratios. The reason to consider the piece size is that it decides how far pieces can be sent over the network. The TTL of HELLO messages is set to its maximum value so that all peers are neighbors of each others. Two sizes of pieces are used whereas the size of the file is left constant. The values considered for the piece size are summarized in table II.

TABLE II
SMALL AND BIG SIZE OF PIECES

| | | |
|---|---|---|
| Small piece | 100 blocks | 1 file = 100 pieces |
| Large piece | 1000 blocks | 1 file = 10 pieces |

Figure 2 plots the average finish time $F_h$ as a function of the number of hops h to the seed for both small and large pieces. Each point in this figure is an average over multiple simulations and over all nodes located at the same number of hops from the seed. As expected, the finish time increases as far as we move away from the source. One can notice in the figure that for small pieces, remote peers have better finish time than for large pieces. This is because the range of transmission of small pieces is longer. A remote peer can then receive more pieces in the choking period and share them with others when pieces are small. The reusability of pieces and network resources improve in this case. This is confirmed in Figure 3 where we plot the average sharing ratio $R_h$ as a function of the number of hops to the seed. Each point in this figure is also an average over multiple simulations and all nodes located at the same number of hops from the seed. It is clear that the sharing ratio for small pieces is more important because distant nodes can now quickly get complete pieces and replicate them in their neighborhood. Unfortunately, this is not the case with large pieces. Large pieces cannot be sent far in the choking period so they propagate in the network as a wave resulting in an under-utilization of network capacities. This is because only one area of the network is in activity at the same time.

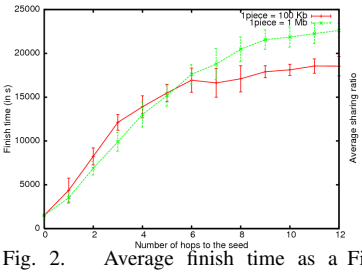


Fig. 2. Average finish time as a function of the number of hops to seed

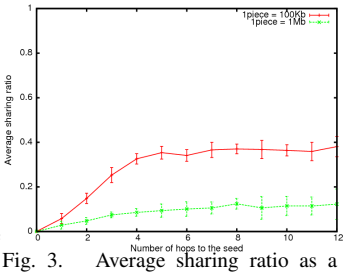


Fig. 3. Average sharing ratio as a function of the number of hops to seed

To better illustrate the idea, we plot in Figures 4 and 5 the sharing ratio for all nodes in the network for small and large pieces. Nodes are ranked and numbered as a function of their distance from the seed. We can see that for large pieces, no sharing between distant nodes exists and that nodes wait for pieces to arrive to their upstream nodes before obtaining them. The use of small pieces make them spread over the network, which reduces the finish time and makes the sharing incentives implemented by BitTorrent work better in wireless ad-hoc networks. Our solution BitHoc supports this modification.

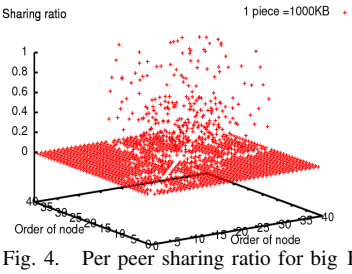


Fig. 4. Per peer sharing ratio for big size of pieces

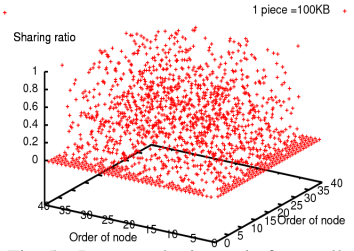


Fig. 5. Per peer sharing ratio for small pieces

## VI. IMPACT OF THE SCOPE OF THE NEIGHBORHOOD

Another important factor in regular topology-unaware BitTorrent is the scope of the neighborhood. In this section, we study the impact of reducing this scope on both the finish time and the sharing ratio. We run several simulations on the topology described in IV-E changing each time the flooding scope (TTL) of HELLO messages destined to peer discovery. Figure 6 compares the finish time for TTL = maximum number of hops, TTL =5, TLL=2 and TTL=1.

Interestingly, the finish time decreases when the neighborhood scope is decreased. This is mainly due to better TCP performance over short paths and to smaller routing overhead. Control packets, namely PIECE UPDATE and HELLO packets, are sent only in the restricted neighborhood. The

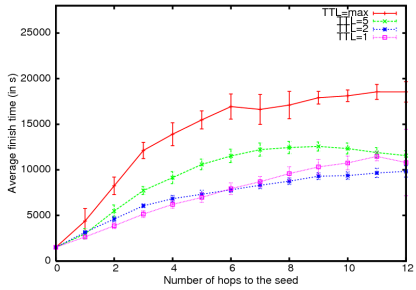


Fig. 6. Average finish time as a function of number of hops to the seed for different flooding scope TTL

case TTL=2 is better than the case TTL=1 because of the interference between physical neighbors. Figure 7 plots the average sharing ratio $R_h$ as a function of the number of hops to the seed for the different values of TTL. Pieces are small in this figure. Results for large pieces are the same. They are not included for sake of space. Unfortunately, we can see that the improvement in the finish time when reducing the neighborhood comes at the expense of a lower sharing ratio. The diversity of pieces in the network decreases and the file propagates more or less as a wave in a unique direction from the seed to the farthest nodes. For small TTL, distant peers

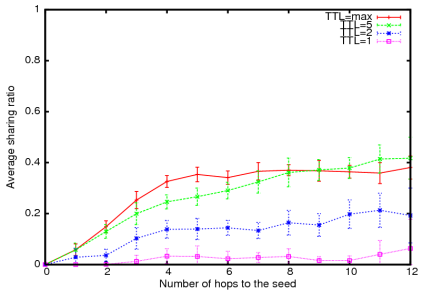


Fig. 7. Average sharing ratio as a function of number of hops to the seed for different flooding scope TTL (small pieces)

can not participate in the replication of pieces. They only wait for pieces to arrive to their physical neighbors to obtain them. Clearly, this is bad for cooperation among peers. An optimal solution is a one that improves the finish time while preserving large values for the sharing ratio.

## VII. BitHoc: Our solution

The main objective of BitHoc is to profit from the advantages of the limited neighborhood, namely the good performance of TCP on short paths, the reduced routing overhead, and the reduced load of flooding control packets. At the same time, BitHoc aims at improving the sharing ratio and the reusability of network resources by creating diversity of pieces in the network. This is done by creating few TCP connections to distant peers. Pieces can then spread over the network and propagate in different directions, which improves the sharing and the download completion time. With BitHoc, several zones of the network can be active simultaneously, which is not the case of the wave generated by regular BitTorrent with limited neighborhood. To this end, we tuned BitTorrent to support the distinction between remote and close peers. The new choking algorithm is aware of the location of peers. It distributes optimistic unchokes between remote and close peers and adds a specific neighbor selection mechanism to select a distant peer. It also applies a new piece selection strategy when the offering peer is distant.

Unlike BitTorrent with limited neighborhood, BitHoc needs a global knowledge about the identifiers of nodes in the network. The TTL of HELLO messages should be set to its maximum value[3]. To distinguish between distant and close peers, each peer maintains two neighbor tables: NEARBY NEIGHBORS TABLE (NNT) and FAR NEIGHBORS TABLE (FNT). When discovering new peers, neighbors whose number of hops is less than or equal to 2 are added to NNT. Other peers belong to FNT. Unlike the HELLO messages, the PIECE UPDATE packets are sent only to neighbors in NNT. Peers do not need to know about all pieces in the network as their piece selection strategy operates only on their nearby neighbors tables. In wireless networks, the replication of pieces is more efficient when it is based on statistics in the close neighborhood since this guarantees a fast local replication compared to when statistics are based on a large neighborhood. As in BitTorrent, when the choking algorithm is executed, three best uploaders are selected as effective neighbors. These three neighbors are chosen from both nearby and far neighbor tables. The peer then serves these three neighbors during the next choking period. But in addition to these effective neighbors, the peer selects a fourth random neighbor from one of the two tables (optimistic slot). The table from which it selects the neighbor is decided by a round robin policy that guarantees an optimal balance between the random unchokes locally and the transmission of pieces to distant neighbors in order to improve diversity. For a succession of optimistic unchokes, the peer selects a peer one time from FNT, q times from NNT and so on. The quantum q is a parameter of BitHoc. It is the ratio of the number of time slots spent on serving nearby neighbors and those for serving far neighbors. It is also the number of slots that a peer should wait before unchoking a distant neighbor again. Our simulations indicate that the choice of this quantum is fundamental in deciding the performance of BitHoc. Furthermore, the strategies of selecting pieces proposed by distant neighbors and selecting effective neighbors from FNT should differ from the ordinary strategies applied by BitTorrent because the objective of BitHoc in unchoking far peers is mainly to improve diversity. The next paragraphs explain the different selection strategies we implement in BitHoc. The following ones study the performance of BitHoc and discuss the choice of the quantum q.

### A. *Selecting a far neighbor at random*

When a regular BitTorrent client decides to optimistically unchoke a peer, it selects it at random with a uniform probability. In wireless networks however, the gain we get from optimistic unchoking in terms of diversity increases with the number of hops. So a peer has more interest in unchoking a

[3]Note that this global knowledge can be obtained by a kind of gossiping without the need to flood the network.

farther peer than another one closer to him. Thus, in BitHoc, to select a far peer to unchoke from FNT, the peer starts by selecting the number of hops with a probability that increases linearly with the number of hops. Let $h_m$ be the maximum number of hops seen by the peer. We suppose that FNT contains only peers at $h_m$ and $h_m - 1$ hops. These are the farthest peers that if we send pieces to them, we are sure of having the largest gain in terms of diversity and reutilization of network resources. It follows that the number of hops is first selected using a probability function p given by this formula:

$$p(h) = \begin{cases} \frac{h}{h_m+(h_m-1)} & \text{if } h \geq h_m - 1 \\ 0 & \text{else} \end{cases}$$

When the number of hops h is chosen, the peer then selects, in a uniform random way, a node among those located at h hops from him as the node to optimistically unchoke.

### B. Selecting a nearby neighbor at random

When the peer needs to select a nearby neighbor, it chooses a node from NNT in a uniform random way. The probability to be chosen is then the same for all nodes. A nearby neighbor is supposed to replicate the pieces it receives in its two-hop limited neighborhood. This replication is fast since the TCP protocol has a good throughput over short paths.

### C. Piece selection strategy when the offering neighbor is far

When receiving a piece offer from a P2P neighbor, the peer checks the number of hops to the offering neighbor. If it is greater than 2, it considers that it is an offer from a far node. In this case, a specific piece selection strategy is applied in order to select the best piece to download from this node. This strategy will be called the *absent piece strategy*. The peer first computes the redundancy of the offered pieces in its close neighbors table and in its piece pool. At the opposite of BitTorrent, the candidate pieces will be those with zero redundancy (no need to download a piece from a distant node if it exists at less than two hops). So a piece can be accepted only if neither the peer nor one of its near neighbors has downloaded it before. In case of multiple absent pieces, one piece among them is chosen in a uniform random way. The absent piece can then be replicated quickly in the near neighborhood. If no absent piece is noticed, the peer sends a REJECT in the piece offer reply packet. In summary, BitHoc supposes that it is better to download a piece existing in the nearby neighborhood from a nearby neighbor. Only absent pieces are taken from far neighbors so as to reduce the routing overhead. This strategy is fundamental for getting good performances with BitHoc.

### D. Piece selection strategy when the offering neighbor is near

Local rarest first is used when the peer receives a piece offer from one of its nearby neighbors. Pieces with the least number of copies in the close neighborhood are selected by this strategy. This is the normal behavior of the standard version of BitTorrent but only applied in the two-hop neighborhood. Here the throughput of TCP is good and the routing overhead is almost inexistent so we can allow ourselves to apply the rarest first policy that guarantees the fast replication of pieces.

### E. Simulation results

To study the performance of BitHoc over the previously described topology, we run several NS-2 simulations. We vary statically the values of the quantum q and observe the behavior of the download finish times of peers and their sharing ratios. Figure 8 compares finish time of ordinary BitTorrent with limited neighborhood (TTL = 2) with BitHoc using different values of the quantum q (q=3, 2 and 1). Each curve presents the average finish time $F_h$ as a function of the number of hops to the seed. Recall that the role of q is to balance optimistic unchokes between close and remote peers. The larger the q is, the smaller is the number of unchokes to remote peers.

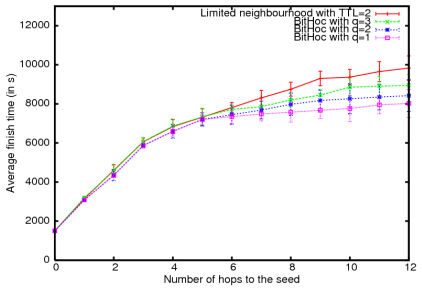


Fig. 8. Average finish time for BitHoc compared to BitTorrent with limited neighborhood

The finish time for BitHoc is better and more equally distributed since far nodes can receive pieces from the beginning of the session and can replicate them in their close neighborhoods. Pieces are sent in an optimal way and in a reduced number so as to keep limited the routing overhead. This creates parallel areas of activity in the network. Far nodes do not need to wait for pieces to arrive to their neighborhoods to download them. Hence, pieces propagate in the network in all directions. This observation is illustrated in Figure 9. The figure compares sharing ratios of ordinary BitTorrent with limited neighborhood (TTL=2) with BitHoc using different values of the quantum (q=3, 2 and 1). Each curve presents the average sharing ratio $R_h$ as a function of number of hops to the seed.

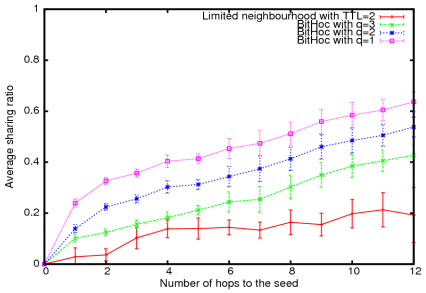


Fig. 9. Average sharing for BitHoc compared to BitTorrent with limited neighborhood

Figure 9 shows that the BitHoc strategies increase considerably the sharing ratios of all peers. This is due to the diversity created by sending original pieces to distant nodes. Unlike the case of limited neighborhood, pieces do not propagate in the

network as a wave but nodes exchange pieces in all directions. This is a major contribution of BitHoc. So, sharing incentives work well in this context and the distribution is less vulnerable to the selfishness of some nodes. The previous results show that a quantum equal to 1 gives a better finish time and a better sharing ratio. Clearly, the performance of BitHoc depends on the value of the quantum q. It has so far been fixed statically and uniformly for all peers. In real deployment of BitHoc, peers should dynamically compute the best quantum to use. The choice of the quantum q is treated in the next sections.

### *F. Optimal static choice of the quantum q*

In this paragraph, we study the static choice of the parameter q. We establish an empiric formula for q and then validate it through simulations. Let $N$ be the number of nodes in the network. Let $h_m$ be the maximum length of a path between two nodes. Let $t$ be the number of pieces in the file and $\alpha_i$ be the number of pieces sent during a choking slot to a node located at $i$ hops. The objective of our balanced optimistic unchoking strategy is to send a copy of each piece to the end of the network and wait for it to return to the middle of the network. Forward and backward pieces meet then in the middle which guarantees the best gain. If there were only one piece in the file, only one seed and only the farthest node is downloading the content, the piece will take $\frac{h_m}{2}$ slots to return to the middle of the network. Now when the file contains several pieces, the node should wait $\frac{\alpha_{h_m}}{\alpha_1} * \frac{h_m}{2}$ before unchoking the farthest node again. It is the number of slots needed for the $\alpha_{h_m}$ pieces to return to the middle of the network hop by hop. Now, if all peers in the network are interested in the content and if we assume nodes to be uniformly distributed in the plane, $\frac{N}{2}$ nodes at maximum can participate in sending pieces to the farthest node. So one needs to increase the waiting time by a factor of $\frac{N}{2}$. So, the static formula for q will be:

$$q = \frac{\alpha_{h_m}}{\alpha_1} * \frac{h_m}{2} * \frac{N}{2} \tag{5}$$

Let us first compute q for the scenario used in the previous sections. In this scenario, $N = 40, t = 100, h_m = 12, \alpha_{h_m} = 1$ and $\alpha_1 = 92$. The values for $\alpha_1$ and $\alpha_{h_m}$ are obtained by simulation. By applying 5, it follows that $q = 1.3$. This is almost the best q recorded by our simulations, which consider only rounded values for q. The same result can be seen in Figure 10 that plots the average finish time as a function of the quantum q. Now, we vary the number of nodes in the network and observe how this impacts the optimal choice of q. We plot optimal q as a function of the number of nodes N when t = 100. Simulations are done on grid topologies of 20 to 100 nodes similar to the topology presented earlier in this paper (10 nodes per row). Figure 11 plots both the computed and simulation results. The values of $\alpha_{h_m}$ and $\alpha_1$ are taken from simulations in both curves. Even though our expression for q is empiric; there is a good match between the two curves. Figure 12 plots the average finish time over all nodes as a function of the chosen quantum q. We do not include simulation results for all values of the number of nodes for sake of space.

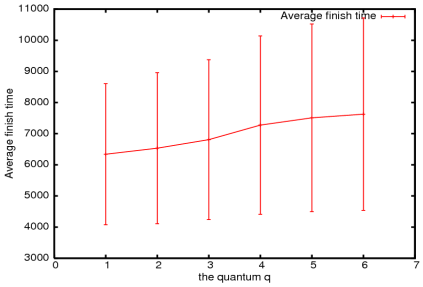


Fig. 10. Average finish time as a function of the chosen quantum

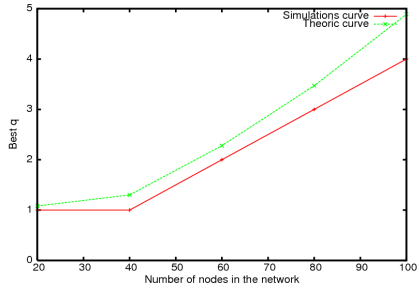


Fig. 11. Best quantum as a function of the number of nodes

Here simulation values of the best q are rounded integer values. Again, the formula $\frac{h_m}{2} * \frac{\alpha_{h_m}}{\alpha_1} * \frac{N}{2}$ describes well the behavior of the optimal q when number of nodes varies. One of the conclusions we can come to is that this quantum increases with N, which means less pieces sent by each peer to remote peers for larger networks.

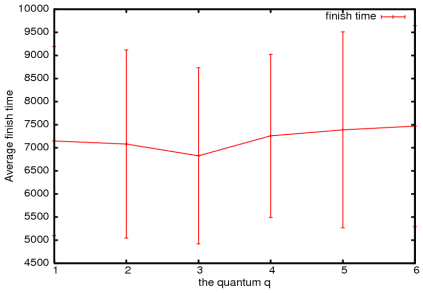


Fig. 12. Finish time as a function of the quantum q

### *G. Dynamic choice of the quantum q*

The topology of the network and its characteristics can change frequently. For instance, $\alpha_{h_m}$ and $\alpha_1$ can vary. Nodes also do not have the same view of the network. So, a further optimization can be made by having q calculated in a distributed and dynamic way by each node. We add this feature to BitHoc. For this, q is initially taken equal to 1. Each node should adapt its value of q as a function of the current values of the parameters:

- $h_m$ is taken equal to the maximum value of the number of hops to a far neighbor. This value changes dynamically. If there is no far neighbor, q is set equal to infinity. (All pieces sent locally)
- The factor $\frac{N}{2}$ in 5 is replaced by the number of neighbors having a number of hops less than or equal to $\frac{h_m}{2}$. This number is noted by $N_{\frac{h_m}{2}}$.

- $\bar{\alpha_i}$: average $\alpha_i$ over the previous $q_d + 1$ choking periods. $\alpha_1$ and $\alpha_{\frac{h_m}{2}}$ are changed in the expression of q by respectively $\bar{\alpha_1}$ and $\bar{\alpha}_{\frac{h_m}{2}}$.

The new formula of q is:

$$q_d = \frac{h_m}{2}.N_{\frac{h_m}{2}}.\frac{\bar{\alpha}_{\frac{h_m}{2}}}{\bar{\alpha_1}} \quad (6)$$

For large networks, a value of $\bar{\alpha}_{h_m}$ equal to zero could be noticed. In this case, one needs to reduce the scope of FNT to only values of number of hops that give $\alpha_{h_m} \geq 1$. Otherwise, the formula will lead to a concentration on far nodes and a big waste of time. Verifying the values of $\alpha_{h_m}$ is very important to achieve the optimality.

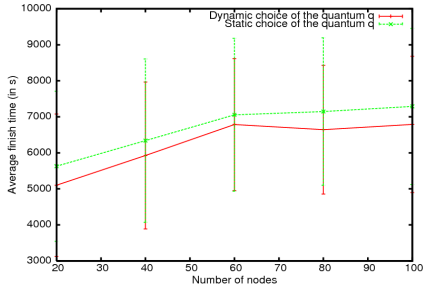


Fig. 13. Finish time for dynamic choice and static choice of q

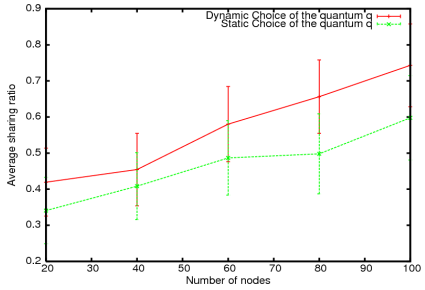


Fig. 14. Sharing ratio for dynamic choice and static choice of q

We rerun some of the previous simulations and we compare the average finish time for static choice of the best quantum q with those of dynamic choice of q. Figure 13 plots the average finish time as a function of the number of nodes for both choices of q. We can observe how far a dynamic setting of q, nodes download the file in a shorter time. We also compare in Figure 14 the sharing ratio for both static and dynamic choice of the quantum. The sharing ratios for dynamic are better than those for the dynamic choice. With this new optimization, peers can find dynamically the best quantum to deploy and adapt their behavior with the changes in network topology and conditions.

## VIII. Conclusions and perspectives

P2P data sharing applications in wireless ad hoc networks should provide good quality of service to their users in terms of finish time and sharing. There is a high potential for these applications but unfortunately, the wireless nature of the network imposes many constraints to be taken into consideration before using regular applications tuned for the wired Internet. Solutions that reduce neighborhood scope allow better finish time than those with random graphs of communications. Nevertheless, limiting the neighborhood is shown, in this paper, to be dangerous in terms of reducing sharing ratios between peers. A final solution must be a trade-off between good finish time and good sharing opportunities. The solution BitHoc that we propose in this paper finds a good management of neighbor and piece selection that reduces finish time and encourages sharing. A peer concentrates on its nearby peers with some connections to far ones. When far neighbors are selected, a special piece selection strategy named absent piece strategy comes into effect. Simulation results show a decrease in service time and a great improve in sharing ratios. Our future work will be on applying our solution to mobile ad hoc networks. High dynamicity of such networks will lead to new interesting problems.

## References

[1] BitTorrent protocol. http://wiki.theory.org/BitTorrentSpecification.
[2] C.Perkins, P.Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers in ACM SIGCOMM'94.
[3] D.B.Johnson,D.A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, Kluwer Academic, 1996.
[4] C.E.Perkins, E.M.Royer, Ad hoc on-demand distance vector routing. In Proc. of the IEEE WMCSA, Feb 1999.
[5] NS: The Network Simulator, http://www.isi.edu/nsnam/ns/
[6] G. Ding, B. Bhargava, Peer-to-Peer File-Sharing over Mobile Ad hoc Networks. In Proc. of the 2nd IEEE PERCOM-W, Orlando, FL, USA, 2004.
[7] Michiardi P., Urvoy-Keller G., Performance analysis of cooperative content distribution for wireless ad hoc networks, IEEE/IFIP WONS 2007, Jan, 2007, Obergurgl, Austria.
[8] The Gnutella specification, 2000, http://dss.clip2.com/GnutellaProtocol04.pdf
[9] The free network project, http://freenetproject.org/
[10] S.Ratnasamy, P.Francis, M.Handley, R. Karp and S.shenker. A scalable content-addressable networks. In Proc. of ACM SIGCOMM, Aug. 2001.
[11] I.Stoica, R.Morris, D.Karger, M.F.Kaashoek and H.Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In Proc. of ACM SIGCOMM, Aug. 2001
[12] B.Y.Zhao, J.D.Kubiatowicz, and A.D.Joseph. Tapestry: an infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley, Apr 2001
[13] A.Rowstron, P.Druschel, Pastry: scalable, diributed object location and routing for large -scale peer-to-peer systems. In Proc. of Middleware, Nov 2001.
[14] L.B. Oliviera, I.G.Siqueira, A.A.Loureiro, Evaluation of ad hoc routing protocols under a peer-to-peer application. In Proc. of IEEE WCNC, Mar 2003.
[15] A.Klemm, C.Lindermann, O.Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In Proc. of IEEE VTC, Oct 2003.
[16] S.M. Das, H.Pucha, Y.C. Hu. Ekta: an efficient peer-to-peer substrate for distributed applications in mobile ad hoc networks. Technical Report TR-ECE-04-04, Purdue University, Jan 2004.
[17] A. Legout, G. Urvoy-Keller, P. Michiardi, Rarest First and Choke Algorithms Are Enough. In Proc. of ACM SIGCOMM/USENIX IMC'2006, Oct. 2006, Rio de Janeiro, Brazil.
[18] A. Nandan, S. Das, G. Pau, M. Gerla, Cooperative downloading in vehicular ad hoc networks. In Proc. of IEEE WONS, Washington, DC, USA, 2005.
[19] S. Rajagopalan, C-C. Shen, A cross-Layer Decentralized BitTorrent for Mobile Ad hoc Networks. In Proc. of ACM/IEEE MOBIQUITOUS, San Jose, CA, USA, 2006.