Mobile-BitTorrent: a BitTorrent Extension for MANETs

Nivia Cruz Quental, Paulo André da S. Gonçalves

Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)
50.740-540 – Recife – PE – Brasil

{ncq, pasg}@cin.ufpe.br

Abstract. *In this paper, we introduce a BitTorrent extension, namely Mobile-BitTorrent, which is specifically designed for MANETs that are composed of resource-constrained mobile devices such as smartphones and PDAs. Mobile-BitTorrent seeks to reduce file completion time by purely exploiting application-layer strategies and the 1-hop broadcast nature of wireless communications. Different from previous studies, Mobile-BitTorrent avoids the use of cross-layering techniques, making implementation simple and adaptable to a wider range of handsets and platforms. Simulation experiments using ns-2 show that Mobile-BitTorrent is able to reduce file completion time by up to 30% compared to that of the classical BitTorrent protocol and in accordance with the scenarios studied.*

1. Introduction

MANETs (Mobile Ad Hoc Networks) are autonomous mobile networks with a multi-hop topology. They are usually formed to work in a short time. The purpose of a MANET is to provide routing functionalities to nodes in a wireless network, with efficiency and robustness. Its multi-hop topology is very flexible, changing with the arrival/departure of nodes and their moves. Most of the P2P (Peer to Peer) networks share all these characteristics, except for the fact that they are mostly designed for wired networks. Among the known P2P protocols in the Internet, BitTorrent [Cohen 2006] has been studied in several research because of its effectiveness in downloading large content in the Internet. Similarities between P2P and MANETs have lead to studies aiming to explore synergies in order to increase performance of P2P applications over MANETs [Hu et al. 2005]. Specifically, recent studies intend to provide a way of bringing the effectiveness of BitTorrent to MANETs, considering the ad hoc architecture and some related scenarios.

A P2P application can be implemented following two approaches: implementing solutions at the application layer, or using cross-layer techniques. In the latter, non-adjacent layers of the protocol stack can communicate to achieve better performances [Conti et al. 2004]. However, the cross-layer approach has some drawbacks. Applications developed using cross-layering are much more complex to install and maintain than solutions implemented exclusively at the application layer. Furthermore, it may lead previously installed applications to behave unexpectedly [Raisinghani and Iyer 2006].

In this paper we propose an extension to BitTorrent, namely Mobile-BitTorrent. This extension seeks to reduce file completion time by only exploiting application-layer strategies and the 1-hop broadcast nature of wireless communications. By doing so, we make Mobile-BitTorrent simple and adaptable to a wide range of current handsets and

platforms. Mobile-BitTorrent introduces specific strategies for selecting and distributing content. It also allows a controlled broadcast of some content messages, taking advantages from the broadcast nature of MANETs. In particular, this paper is concerned with scenarios like talks and meetings in which MANETs are formed and where a node makes content available and all nodes are interested in the same content.

This paper is organized as follows: Section 2 briefly describes the BitTorrent protocol. Section 3 presents the Mobile-BitTorrent protocol. Related work are presented in Section 4. Section 5 evaluates through simulation the performance of BitTorrent and Mobile-BitTorrent, making comparisons. Finally, Section 6 provides concluding remarks.

2. The BitTorrent Protocol

BitTorrent [Cohen 2006] is a popular protocol used for P2P file sharing on the Internet. It splits the file at issue into small chunks called *pieces*. Each piece is further split into *blocks*. By dividing the file in this manner, BitTorrent facilitates peers to download different chunks of the file from multiple sources at the same time. The size of both pieces and blocks is fixed and implementation-dependent. Each piece has an index that allows peers to identify to which part of the file it belongs. A block is identified by its offset within a piece. A peer which has all the pieces of the file is called a *seed*. A peer which is downloading pieces is called a *leecher*.

When a peer is interested in a file, it sends a request to a centralized entity called *Tracker* in order to discover other peers that are interested in the same file. Peers involved in the same download communicate with each other using messages transmitted through a TCP connection. The main messages in the BitTorrent protocol are described as follows:

- `HANDSHAKE` - sent to perform an initial handshake between two peers.
- `BITFIELD` - its payload is a bitfield representing the pieces that have been successfully downloaded.
- `INTERESTED` - informs a peer that a peer is interested in its blocks.
- `REQUEST` - requests a block specifying its offset.
- `PIECE` - contains a block and informs its corresponding offset and piece index.
- `HAVE` - informs that a peer has just completed a piece.
- `CHOKE` - informs a peer that it cannot request pieces from the sender of this message.
- `UNCHOKE` - informs a peer that it can request pieces from the sender of this message.

Figure 1 shows an example of message flow in BitTorrent. ***Peer 1*** sends a `HANDSHAKE` message to ***peer 2*** to establish a connection and ***peer 2*** responds with the same message. Thus, ***peer 1*** sends a `BITFIELD` message to show to ***peer 2*** which pieces are available for upload and vice-versa. If one wants to receive content from another, it must send an `INTERESTED` message. Let us consider that ***peer 1*** is interested in pieces from ***peer 2***. In this case, ***peer 2*** must check if ***peer 1*** is allowed to download. If so, it must be unblocked using the message `UNCHOKE`. Now ***peer 1*** is authorized to download content from ***peer 2***, it sends a `REQUEST` message to request a block, specifying the piece index and the block offset. Finally, ***peer 2*** sends a `PIECE` message, containing the desired block. When ***peer 1*** finishes a piece, it sends a `HAVE` message to everyone in the swarm informing that a new piece is available for upload.

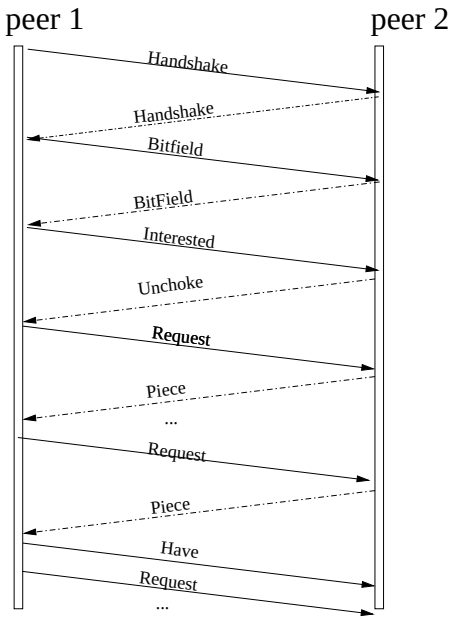

Figure 1. Example of message flow in BitTorrent

3. Mobile-BitTorrent

Mobile-BitTorrent seeks to improve download performance by only exploiting application-layer strategies and the 1-hop broadcast nature of wireless communications. In Mobile-BitTorrent there are two kinds of peer: *disseminator peer* and *common peer*. The former is a seed that periodically sends different `PIECE` messages via broadcast. However, its 1-hop neighbors cannot forward these messages. Furthermore, the disseminator peer also responds to `REQUEST` messages received over TCP connections. The latter does not distribute `PIECE` messages through broadcast but it can use `PIECE` messages received in both unicast and broadcast modes to complete the download. In addition, a *common peer* is somewhat independent from the disseminator since it can still request content via unicast and respond to `REQUEST` messages received over the TCP connection.

Mobile-BitTorrent introduces an application-layer interface, namely BMI (*BitTorrent Mobile Interface*), that manages traffic to be sent via broadcast by the disseminator peer and received by one or more common peers. In the next sections we detail this interface and describe the following: how to select messages to be sent in broadcast, how to select the disseminator peer, and how frequently the disseminator broadcasts `PIECE` messages.

3.1. BMI

BMI (*BitTorrent Mobile Interface*) is an application-layer interface between the BitTorrent core and the transport-layer. It manages incoming/outgoing broadcast messages. Both disseminator peers and common peers must have BMI implemented. Since TCP is a point-to-point protocol, all the messages to be sent via broadcast must be encapsulated into UDP segments. Figure 2 shows the communication protocol stack used in Mobile-BitTorrent. On the sender side, BMI receives a `PIECE` message from the application for transmission. This message must carry some identifier of the content (*e.g.*, a hash of the file shared). Then, BMI sends the message to the UDP socket to which it is attached,

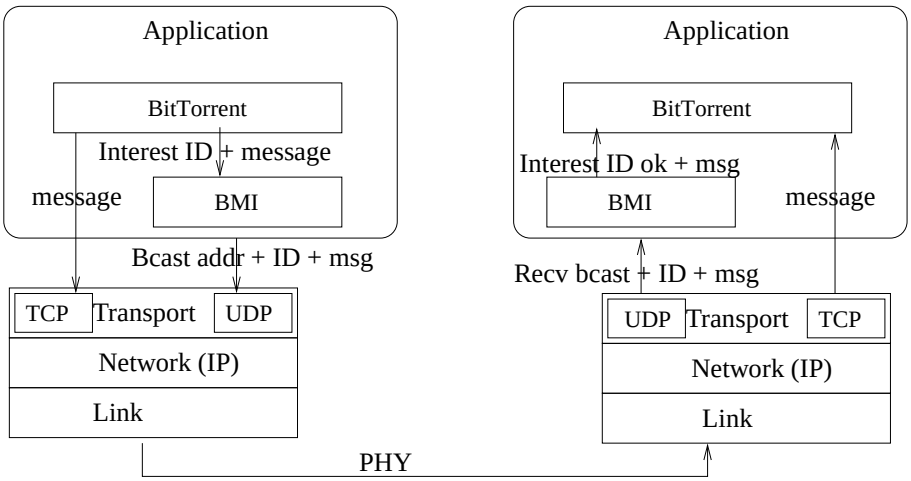

Figure 2. Communication protocol stack and the BMI interface

setting the network broadcast address as destination. On the receiver side, BMI acts as a filter. If the peer is interested in the content, the BMI de-encapsulates the original `PIECE` message and forwards it to the application. Otherwise, *i.e.* if the peer is not interested, the BMI silently discards the message received.

3.2. Selecting messages for broadcasting

The choice of which `PIECE` messages must be sent in broadcast is done by considering the block information. The disseminator peer uses a strategy to select and broadcast blocks. Such strategy is different from that used by common peers to request blocks in unicast mode. The disseminator peer gives priority to the rarest blocks of the rarest pieces, following the ascending order of piece index. Common peers request blocks in sequence from the rarest pieces, following the descending order of piece index. The disseminator peer never sends the same block more than once.

3.3. Selecting the disseminator peer

The *disseminator* selection follows the criterium of seed with the greatest identifier (eg. the IP address). It can be done by using a leader election algorithm for MANETs, like those presented in [Dagdeviren and Erciyes 2008], [Malpani et al. 2000]. Seed information can be obtained through incoming `BITFIELD` messages.

3.4. Distribution Periodicity

Message broadcasting must be done carefully to avoid the increase of packet collisions and in turn the degradation of the download performance. The disseminator peer broadcasts `PIECE` messages periodically. The time interval between consecutive transmissions is uniformly chosen at random on the interval [0, M] seconds. We consider M as a design parameter. Determining the best M value is out of the scope of this work.

4. Related Work

Rajagopalan *et al*. [Rajagopalan and Shen 2006] use cross-layering techniques to integrate application functionalities to the ANSI routing protocol. Krifa *et*

al. [Krifa et al. 2009] present BitHoc, a cross-layer solution, which offers a component for peer management and content sharing based on the current MANET topology. BitHoc use information from the OLSR (Optimized Link State Routing) routing protocol to update information of available peers. Souza *et al*. [Souza and Nogueira 2008] explore scenarios similar to that studied in this paper. They propose a set of modifications in BitTorrent to deal with spatial-temporal locality. These modifications use resources from several layers in a cross-layer design. In this approach, peers are organized in clusters with a leader. The leader downloads the content and forwards it to its peers in the cluster in multicast mode. These peers are disconnected from the BitTorrent network and wait for the content came from the leader.

Mobile-BiTorrent differs from the later approaches, because it does not change the protocol stack architecture and does not depend on the MANET underlying routing protocol. Mobile-BitTorrent, though, keeps the autonomy of peers to request and receive content from any peer, despite of Mobile-BitTorrent also uses a leader in the network to distribute content. Additionally, in this paper we include a mobile scenario, not covered in [Souza and Nogueira 2008].

5. Performance Evaluation

In this section, we compare the performance of the Mobile-BitTorrent protocol to that of the classical BitTorrent protocol. We intend to evaluate the efficacy of the Mobile-BitTorrent strategies in a MANET. In particular, we are interested in evaluating the download performance without interferences from the peer discovering mechanism. So that, we disregard the communication overhead with the Tracker. We also consider that the seed with the greatest identifier is the *disseminator*. The simulations were performed with ns-2 [Fall and Varadhan 2007]. To run simulations, we extended ns-2 with the BitTorrent module developed in [Eger et al. 2007] and the OLSR module developed in [Paquereau and Helvik 2006]. In addition, we implemented the necessary changes in the classical BitTorrent protocol to support simulation experiments with the Mobile-BitTorrent protocol. The PHY and MAC layers were modeled by following the IEEE 802.11g specification.The communication range of ad hoc nodes was set to *50 m* to simulate an indoor environment. We are particularly interested in studying situations such as local events, classes, talks, and conferences. The simulation area was set to *150m* x *150m*. The *Two Ray Ground* was the propagation model used in simulations. In our simulation study, we evaluated the following performance metrics:

Number of segments - the total number of transport-layer segments generated until download is finished by every peer.
Ratio of segments - the ratio of the total number of UDP/TCP segments sent to the total number of segments sent.
Number of lost packets - total number of packets lost until download is finished by every peer.
Download time - time in order to a peer to complete the download.
Number of routing packets - total number of routing messages sent, including transmissions between hops.
End-to-end delay - time taken for transport-layer segments to be transmitted across the network from source to destination.

We consider two simulation scenarios: a fixed scenario and a mobile scenario. In the former, ad hoc nodes are arranged in a grid topology. Each node is *1 m* apart from its neighbors in a row. In the latter, nodes are initially arranged as in the fixed scenario but they move in accordance with a modified *Random Waypoint* model, which works as follows: after a pause time, nodes move to a random destination in the simulation area. When reaching this destination, they stop and wait for the same pause time. Next, they start moving again, and so on. The pause time is set to 10 s. Both mobile and fixed scenarios share some characteristics, which are listed below:

- Each node executes only one instance of the application.
- Simulation starts with a single seed.
- Nodes join the network at random times between 0 s and 1 s after simulation starts.
- During download, we do not simulate churn.
- When a peer finishes the download, it stays in the network to cooperate with other peers.
- The size of the file to be shared among peers is 100 MB; the piece size is 512 KB and the block size is 16 KB.
- The design parameter M used in Mobile-BitTorrent is set to $1s$. This value was obtained based on preliminary evaluation of several values, where $1s$ was taken as the most suitable for the experiment [Quental 2009].
- A simulation finishes when every peer concludes the file download.

We adopt a 99% confidence level to present all results in this section. The 99% confidence intervals are represented by error bars in the next Figures in which each point is the mean of results from 20 simulations.

5.1. Fixed Scenario

In this scenario, the performance metrics are evaluated in function of the number of peers. We study cases for 4 (2x2), 9 (3x3), 16 (4x4), 25 (5x5), 36 (6x6), and 49 (7x7) *peers* in the network.

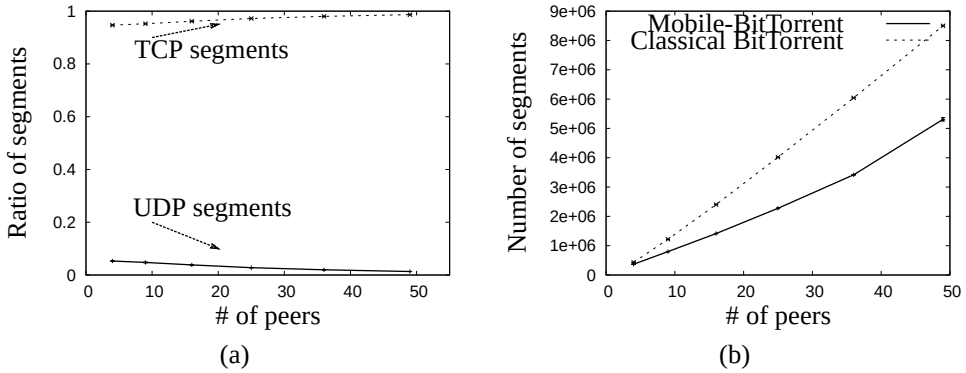

Figure 3. Scenario without mobility: (a) Ratio of segments (b) Number of segments

Figure 3(a) shows the ratio of TCP and UDP segments when using Mobile-BitTorrent. Note that, for 4 *peers*, only 5.4% of all segments are sent in broadcast mode.

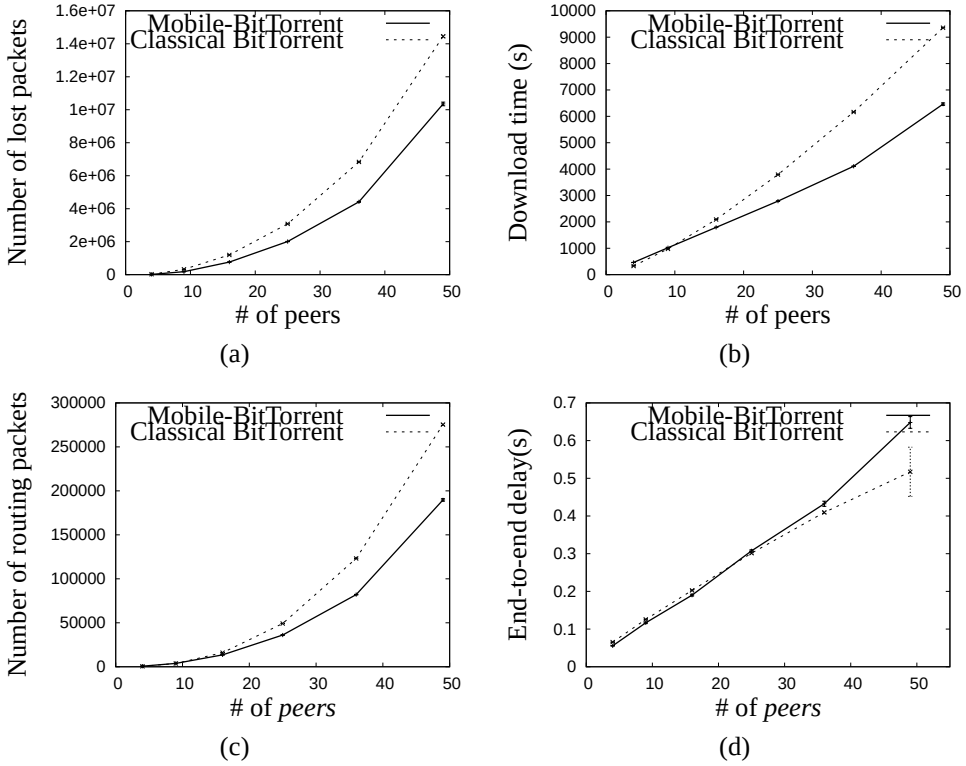

Figure 4. Scenario without mobility: (a) Number of lost packets (b) Download time (s) (c) Number of routing packets (d) End-to-end delay(s)

As the number of peers increases, this value decreases, reaching 1.3% for 49 peers. This reduction occurs because as the number of peers increases, there are more active TCP connections generating additional unicast traffic. Meanwhile, the amount of broadcast traffic does not vary since there is a single disseminator.

Figure 3(b) shows that Mobile-BitTorrent has reduced overhead of segments compared to BitTorrent. For 36 peers, in particular, Mobile-BitTorrent generates 50% less segments than the classical BitTorrent. The distributed content serves all peers around the disseminator. Thus there is a reduction of generated requests for blocks in the network.

Figure 4(a) shows that the amount of segments lost with Mobile-BitTorrent on the top of the MANET is up to 40% smaller than that of the classical approach. This occurs due to the reduced amount of segments that Mobile-BitTorrent generates and sends across the network.

Figure 4(b) shows download times on both the classical BitTorrent and the Mobile-BitTorrent. Note that up to 9 peers, there is no significant difference between these approaches. However, starting from 16 peers, Mobile-BitTorret downloads the file faster than BitTorrent. For 49 peers, the download time on Mobile-BitTorrent is 30% faster than that of BitTorrent. Again, this is due to the reduced amount of segments necessary to delivery content with Mobile-BitTorrent.

Figure 4(c) shows that the use of Mobile-BitTorrent contributes to reduce the routing overhead. In particular, for 49 peers, this overhead is 20% smaller than that of the classical approach. A reduced routing overhead using Mobile-BitTorrent is expected be-

cause it provides peers with better download times. So that, much less routing messages can be sent until every peer has finished the download.

Figure 4(d) shows that the end-to-end delay is similar using both approaches for a number of peers varying from 4 to around 36 . However, the end-to-end delay for Mobile-BitTorrent is slightly greater than that of the classical BitTorrent when the number of peers is greater than 36. The reason is as follows: the more the number of peers is increased, the more collisions increase as a side effect of the broadcasting. Even though, download times on Mobile-BitTorrent are still lower than those on BitTorrent because of the lower number of segments sent across the network.

5.2. Mobile scenario

In this scenario, the performance metrics are studied in function of the speed of 25 nodes, varying from *0.0 m/s* to *1.5 m/s* in steps of *0.5 m/s*.

Figure 5(a) shows that, on average, 2.8% of the total number of segments is generated from content sent to the BMI when the speed of the nodes is set to *0.0 m/s*. On the other hand, the more speed increases, the more this average value reduces, achieving 2.0% for a speed of *1.5 m/s*. This occurs because the amount of TCP traffic increases as the speed of the nodes grows while the amount of traffic sent via broadcast does not change. Thus, less nodes take advantages from dissemination.

Figure 5(b) shows that the overhead of segments with Mobile-BitTorrent is significantly lower than that of the BitTorrent. In the worst case, when there is a slight rising caused by TCP traffic, this overhead is 20% lower than that of the classical approach.

Figure 6(a) shows that BitTorrent provides higher packet losses. In particular, the more the speed of the nodes increases, the more the total number of lost packets decreases, no matter which approach is used. The reason is as follows: remember that at the beginning of a simulation experiment, the nodes are in close vicinity so that they share the same collision domain. When the speed of the nodes is increased, more fast they are spread over the simulation area. This reduces the number of competing nodes in a same collision domain more rapidly, reducing the number of packet collisions and in turn the number of lost packets in the network.

For both approaches, Figure 6(b) shows that the routing overhead steeply increases when the speed of the nodes goes from $0.0\ m/s$ to $0.5\ m/s$. This occurs because OLSR

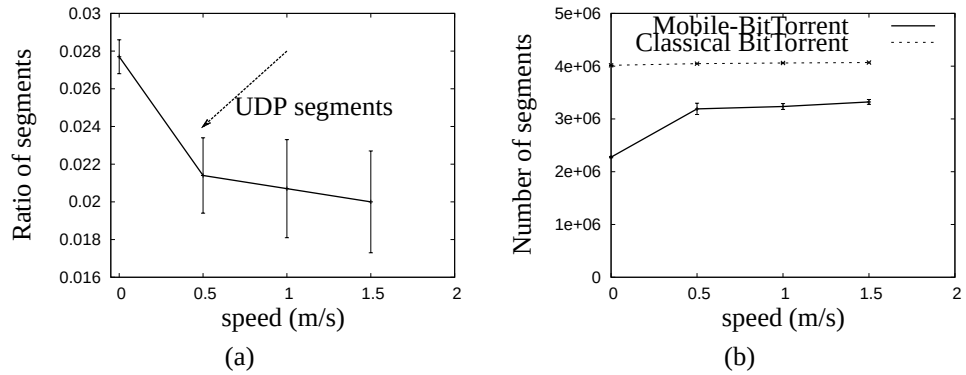

Figure 5. Scenario with mobility: (a) Ratio of segments (b) Number of segments

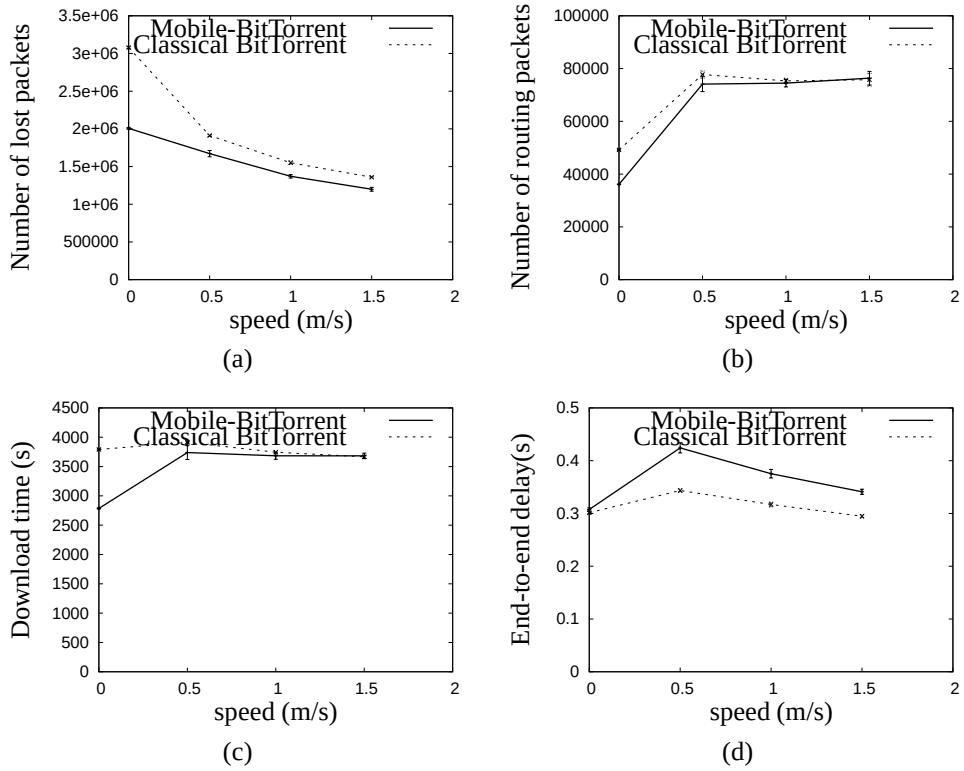

Figure 6. Scenario with mobility: (a) Number of lost packets (b) Number of routing packets (c) Download time (s) (d) End-to-end delay (s)

reacts to topological changes by reducing the maximum time interval for periodic control message transmission. In addition, the routing overhead for both approaches is similar and nearly constant for the following speeds: $0.5\ m/s$, $1.0\ m/s$, and $1.5\ m/s$. This happens because, in such cases, both approaches exhibit similar download times (see Figure 6(c)), which in turn make the routing protocol to transmit nearly the same amount of periodic control messages during the download.

Figure 6(c) shows that download times on BitTorrent do not vary significantly as the speed of the nodes grows. On the other hand, download times on Mobile-BitTorrent increase when changing the speed of the nodes from $0.0m/s$ to $0.5m/s$. Such behavior can be explained as follows: at the beginning of a simulation experiment, the nodes are in close vicinity so that the broadcast content from the disseminator can reach every other node. After nodes starting moving, the number of nodes that benefit directly from the broadcast content is reduced. As a side effect, more unicast traffic is generated, which increases download times. Even though, download times on Mobile-BitTorrent are less or equal than those on BitTorrent.

Figure 6(d) shows that the end-to-end delay with respect to both approaches increases when the speed of the nodes changes from *0.0 m/s* to *0.5 m/s*. There are two reasons for such behavior: first, additional delays are incurred for repairing broken routes due to mobility. Second, since OLSR increases its control traffic overhead to deal with mobility, this introduces additional delays to packets reach destination. With respect to the increase of the speed of the nodes step-by-step from $0.5\ m/s$ to $1.5\ m/s$, we note that end-to-end delay slightly reduces as the speed of the nodes increases. This result is

expected because packet losses are also reduced as the speed of the nodes increases (see Figure 6(a)). It is worth noting that although the end-to-end delay is greater when using Mobile-BitTorrent, download times on Mobile-BitTorrent are shorter than or, in the worst case, equal to those on BitTorrent.

6. Conclusions

Mobile-BitTorrent is an alternative to the use of cross-layer based implementations on resource-constrained mobile devices. Mobile-BitTorrent adopts specific application-layer strategies for selecting and distributing content. Simulations results showed that distributing through broadcast a small amount of data segments (1.3% to 5.4%) can effectively enhance performance. In conclusion, our study suggests that is possible to improve the performance of BitTorrent over MANETs by using purely application-layer strategies.

References

Cohen, B. (2006). BitTorrent Specification. Technical report, BitTorrent.org.

Conti, M., Maselli, G., Turi, G., and Giordano, S. (2004). Cross-Layering in Mobile Ad Hoc Network Design. *IEEE Computer*, 37(2):48–51.

Dagdeviren, O. and Erciyes, K. (2008). A hierarchical leader election protocol for mobile ad hoc networks. *Computational Science–ICCS 2008*, pages 509–518.

Eger, K., Hossfeld, T., Binzenhofer, A., and Kunzmann, G. (2007). Efficient Simulation of Large-scale P2P Networks: Packet-level vs. Flow-level Simulations. In *Proceedings of the 2nd UPGRADE-CN*, pages 9–16, New York, NY, USA.

Fall, K. and Varadhan, K. (2007). NS Notes and Documentation. Technical report, The VINT Group.

Hu, Y. C., Das, S. M., and Pucha, H. (2005). Peer-to-Peer Overlay Abstractions in MANETs. In Wu, J., editor, *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, volume 1, chapter 47, pages 857–874. CRC Press.

Krifa, A., Sbai, M. K., Barakat, C., and Turletti, T. (2009). BitHoc: A Content Sharing Application for Wireless Ad hoc Networks. In *Proceedings of the IEEE Percom*.

Malpani, N., Welch, J. L., and Vaidya, N. (2000). Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th DIALM*, pages 96–103.

Paquereau, L. and Helvik, B. E. (2006). A Module-based Wireless Node for NS-2. In *Proceedings of the WNS2*, page 4, New York, NY, USA.

Quental, N. C. (2009). Um Sistema de Disseminação de *Pieces* para a Melhoria do Desempenho de Aplicações BitTorrent sobre MANETs. Master's dissertation, CIn - UFPE.

Raisinghani, V. and Iyer, S. (2006). Cross-layer feedback architecture for mobile device protocol stacks. *IEEE Communications Magazine*, 44(1):85–92.

Rajagopalan, S. and Shen, C.-C. (2006). A Cross-layer Decentralized BitTorrent for Mobile Ad hoc Networks. In *Proceedings of the 3rd MobiQuitous*, pages 1–10.

Souza, C. and Nogueira, J. M. (2008). Um Estudo do BitTorrent em Redes ad hoc sem Fio Críticas com Localidade Espaço-temporal. In *Anais do 25º SBRC*, pages 329–342.