Performance analysis of cooperative content distribution in wireless ad hoc networks

Pietro Michiardi
Institut Eurecom
Pietro.Michiardi@eurecom.fr

Guillaume Urvoy-Keller
Institut Eurecom
Urvoy@eurecom.fr

Abstract— **In this paper we focus on the problem of content distribution in wireless ad hoc networks. Our goal is to study the performance of a cooperative content distribution mechanism to distribute content from one source to a potentially large number of destinations. Despite the large literature on content distribution schemes available for wired settings we argue that the very nature of the underlying ad hoc network poses new challenges that cannot be addressed with current schemes. We propose a cooperative peer-to-peer scheme that allows parallel download of the content based on swarming protocols. Our scheme builds a distribution overlay network that takes into account traffic locality and allows peers to trade parts of the content while sustaining cooperation. We evaluate through simulations the performance of our scheme for different static scenarios using a variety of metrics to characterize the impact of our solution at different layers of the system stack. Our results highlight the great benefits of our solution in terms of system fairness, achievable throughput, and energy consumption. We also study the scalability properties of our solution under the extended network model.**

I. Introduction

The goal of this paper is to study whether the advantages of cooperative (peer-to-peer) content distribution as seen in the Internet can carry over in ad hoc networks. To do this, we develop an application layer content distribution scheme and we study its performance extensively. Recent works have mainly focused on the content lookup problem and on the creation of multicast wireless overlays (*e.g.*, application layer multicast) in mobile ad hoc networks. In contrast, in this paper we study the performance achievable at the application layer by a content distribution scheme in a static ad hoc network, neglecting the content lookup problem. Though node mobility represents a challenging problem that needs to be addressed to achieve efficiency in a mobile environment, in our work we are interested in the issues caused by node interference and the multi-hop nature of the underlying network. Indeed, recent works [11], [15] show that the throughput achievable in an ad hoc network is inversely proportional to the route length of data flows: traffic locality determines to a large extent the capacity scaling of ad hoc networks. The seminal work [13] also shows the role of interference on performance degradation of a static ad hoc network.

We address these issues using cooperative content distribution (CCD) in which a mesh of cooperating peers download the content in parallel. In a CCD scheme the content is split into pieces available at one or more sources (that we call seeds). The mesh overlay, which also includes all the available seeds, is constructed on an on-demand basis by peers (that we call leechers) interested in downloading the content. Peers trade pieces they may hold until all pieces of the content are available at every peer. We focus on CCD schemes based on a distributed piece scheduling algorithm wherein scheduling decisions are based on the statistical distribution of pieces in the mesh overlay.

In this paper we propose an overlay construction mechanism that mitigates the issues raised by the underlying ad hoc network by favoring local interactions among the peers. On top of the on-demand overlay, peers involved in the content distribution execute a piece scheduling algorithm inspired by the BitTorrent protocol [1]. We study the impact of the overlay structure on the performance of the CCD scheme and show the improvements of our overlay construction technique over a randomized approach. We also study the piece propagation process and the number of parallel connections (thus the spatial reuse) that we can achieve. Our measurements are taken at different levels of the protocol stack: we show the fairness of the CCD scheme that characterizes energy consumption and investigate on potential issues due to message retransmission both at the MAC and at the transport level. Finally, we study the overhead and the scalability of our protocol.

The remainder of the paper is organized as follows: in Section II we detail our CCD scheme, specifying the overlay construction mechanism and the content distribution protocol. In Section III, we define our simulation-based study and present detailed results of our investigation. We conclude in Section VII where we also point out our future work.

II. The protocol

In this section we describe the building blocks of our CCD scheme: the overlay construction mechanism and the piece scheduling algorithm. We also describe an additional algorithm used by peers to foster collaboration. An extended description of our protocol can be found in [18].

A. Overlay construction

The goal of the overlay construction scheme is to build logical connections among peers while mitigating the problems due to the underlay network. Our distributed overlay mechanism is executed both by seeds and leechers on an on-demand basis.

The core objective of our mechanism is to guarantee a maximal number of close neighbors, a parameter that plays a crucial role for limiting interference and long underlay routes: indeed, the distance in underlay hops between two overlay neighbors should be minimal to achieve good performance [19]. Moreover, our scheme guarantees logical connectivity, trading off on the search horizon used to contact potential overlay neighbors.

For each peer p_i, we define the size k_i of the set of overlay neighbors a peer wants to achieve. A peer uses the expanding ring search technique and uses the size k_i as a stop criterion. In practice, peer p_i first obtains its physical neighborhood from the routing layer: with few exceptions all ad hoc routing protocols have an initial neighbor discovery phase that produces 1-hop neighborhood information. An application-level broadcast message is then propagated in the network in order to search for peers (both seeds and leechers) interested in a particular content. We call this message an overlay request (OREQ) message, which contains: the p_i identifier, a content identifier, a sequence number and a TTL field. The p_i identifier can be for example the corresponding node identity, such as its IP address. The content identifier can be for example a message digest computed over the filename of the content. The TTL field is initially set to one. The sequence number is used to discard duplicate query messages.

Upon reception of a OREQ message, if a peer p_j is downloading the requested content, if p_j has not reached her target of k_j neighbors, and if p_i is not an overlay neighbor of p_j, it replies with a overlay reply (OREP) message. The OREP message contains p_j identifier, and is unicast back to the requestor p_i. Note that neighborhood relationships are symmetrical: if p_j is in the logical neighborhood of p_i, then the opposite is also true. If the TTL in the OREQ message is greater than zero, p_j decrements its value by one and broadcast the modified OREQ message to its neighbors. The process continues until TTL=0.

The OREQ originator collects OREP messages and checks the size of its overlay neighborhood, breaking ties uniformly at random when the size is greater than k_i. If the current size is less or equal to k_i the originator increments the TTL field by one and the process starts over, until the target k_i has been reached.

We extend this method by introducing an exception to the size criterion. The TTL field cannot be increased arbitrarily, but has a maximum value MAX_{TTL}. If this value is reached, the expanding ring search algorithm stops. Note that MAX_{TTL} depends on content popularity; *e.g.*, high values of MAX_{TTL} are required for an unpopular content, as peers holding pieces of the content might be far away.

Our search process requires a stop criterion based on two parameters that we empirically tune in our simulations. In our experiments we set $k_i = k \forall p_i$; furthermore, we focus on the extreme case of a very popular content: all nodes in the network except the seed are leechers. We believe this scenario to be the most stressful for the underlay network and in this work we study the performance of our CCD scheme in such context.

B. Piece scheduling algorithm

The piece scheduling algorithm allows a peer to decide which pieces it wants to receive from remote peers. Our algorithm is decentralized and based on local information: peers use the statistical distribution of pieces in their overlay neighborhood to determine which piece is best to replicate. In this work we borrow the heuristic adopted by BitTorrent [1]: rare pieces, that is pieces that have the lowest number of replicas in the neighborhood, are selected for download. A downloading peer selects among the set of rare pieces the one it is missing. This technique is known as the local rarest first (LRF) piece scheduling and it has been shown to surmount the last piece problem [16]. The ultimate goal of LRF is to enforce diversity in the piece distribution process, avoiding the undesirable case of a single peer holding a rare piece thus becoming a bottleneck for the system [16]. Recent works ([4], [5], [25]) have shown the benefits of LRF over a random scheduling approach.

In our scheme, only overlay neighbors trade pieces of the content, hence peers need to exchange only a limited amount of information on the pieces they hold that depends on the neighborhood size k_i of a peer. We assume pieces to be identified by a numerical value, ranging from 1 to the number of pieces the content is split into. Once the overlay construction phase has completed, a peer p_i performs an handshake transaction with all peers in its logical neighborhood. During the handshake, peers exchange a bit vector in which an element set to one indicates that the corresponding piece is available at that peer. The bit vector's size is equal to the number of pieces the content is broken into.

Subsequently, p_i incrementally informs its neighborhood on the piece it downloaded during the distribution process using a small control message of size $1byte$ with the identifier of the newly available piece. We adopt the suppression technique described in [1]: control messages are only sent to peers that do not have the indicated piece.

Note that the information on the piece statistical distribution is delicate: a loss of a control message can alter the normal execution of the scheduling algorithm leading to serious inefficiencies, such as the creation of bottleneck peers. As opposed to [19], we use a reliable transport mechanism (TCP) to exchange control information as well as for piece exchange.

C. Peer selection algorithm

We now briefly describe an additional algorithm that we use to foster peer cooperation. Our technique is inspired by the BitTorrent protocol and has the goal of guaranteeing a reasonable level of upload and download reciprocation.

While a peer schedules the next piece it wants to download, we allow the peer holding the requested piece to decide whether the request will be fulfilled. The selection is based on a rate-level tit-for-tat strategy: peers that previously interacted with the decision maker are ordered based on the service capacity they offered. Only those peers that are in the top

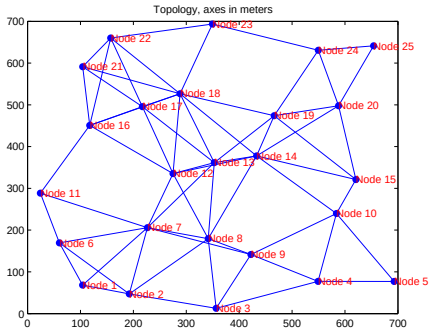


Fig. 1. Topology 1: dense network with 25 nodes.

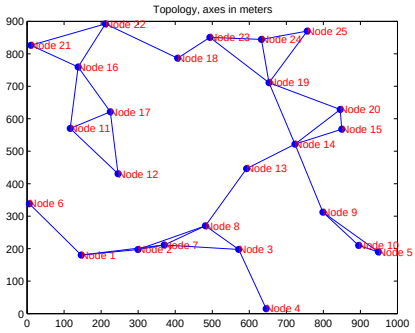


Fig. 2. Topology 2: sparse network with 25 nodes.

3 positions will be served back. Our algorithm admits one exception: one additional neighbor is selected randomly in the overlay neighborhood regardless of the service capacity it offered. This is done in order to allow peers to bootstrap the download process when they have no pieces to trade; furthermore, it allows a peer to seek a large overlay neighborhood for peers with a high service capacity. This algorithm is also executed by seeds. However, instead of giving priority to peers with a high download capacity, seeds uniformly distribute pieces to leechers, as documented in [16].

Note that our algorithm is executed locally: peers only need to keep statistics on the download rate perceived from other peers they interacted with. We do this in our implementation by using a sliding window of size $10sec$.

III. Performance evaluation

In this section we examine the impact on the performance of our CCD scheme of the main system parameters. Furthermore, we study the scalability properties of our scheme when the number of peers in the network increases reasonably.

A. Simulation set-up

Our evaluation is done using the *Qualnet* [3] network simulator. In our simulations we use the CSMA/CA 802.11a MAC protocol and use the RTS/CTS-Data/ACK mechanism. We set the data rate at 36Mbps, which leads to a $230m$ data radio range in free-space. We use the unicast proactive ad hoc routing protocol OLSR [2]. Data packets and signaling messages are sent using the TCP transport protocol while messages for the overlay construction are sent using UDP.

If not otherwise stated, we consider ad hoc networks formed by $N = 25$ nodes. We study the performance of our scheme under two scenarios. The first is characterized by a dense network wherein interference issues are exacerbated: Figure 1 shows an instance of nodes uniformly deployed over a $700m^2$ square area. In the second scenario we emphasize the effects of multi-hop routes with a sparse network: Figure 2 shows an instance of nodes uniformly deployed over a $1000m^2$ square area

In our experiments the size of the content is set to 5MB. and is split into pieces of 16 blocks, each block being of size 16KB. The choice of piece and block sizes follow the heuristic proposed by BitTorrent and by the work presented in [19]. We run several experiments to understand the impact of different piece and block sizes: due to space limitations we do not present these results as the impact on performance is negligible.

We assume peer arrival rates to be representative of a *flashcrowd* scenario: peers bootstrap the content distribution overlay and fetch the content at the same time. In our experiments we focus on a popular content: we have one seed and all other nodes in the network are leechers.

Each point in the following plots is the average result over 5 independent simulation runs for every simulation setting.

B. Performance metrics

The first metric we use to evaluate the performance of our scheme refers to the time to distribute the content to all receivers. We define the *time to download* metric (TTD) that indicates the time at which a peer received the whole content and study its cumulative distribution function (CDF). In Section IV-B we use the following additional performance metrics:

- *Piece propagation*: for every leecher in the system, we study the reception time of the first and the last piece of the content[1]. This allows to deduce the speed at which pieces propagate in the network at two significant time phases and to observe any variation of that speed in the distribution process.
- *Aggregate upload capacity*: we evaluate the service capacity of our system by summing up the upload rates for every peer. Upload rates are obtained by summing the total bytes uploaded by a peer in $10sec$.
  This metric allows to deduce the efficiency of our CCD mechanism.
- *Number of parallel downloads*: the number of concurrent transmissions in the system is calculated by summing up the number of unique peer identities involved in the reception of a piece in $10sec$.

[1]Note that piece are not ordered: the first/last piece received does not correspond to first/last piece index.

The number of parallel transmissions is related to the spatial reuse that we achieve in the network due to the distribution overlay structure.

The choice of $10sec$ time intervals is determined by the periodicity ($10sec$) used by peers to decide to serve pieces, as described in Section II-C.

In Section IV-C we discuss on the performance of our scheme using measures taken at different layers of the protocol stack. We define an *energy consumption* metric that indicates the energy consumed at the physical layer. The energetic model implemented in Qualnet follows the one presented in [9]: only the transmission and reception of data consumes energy while no energy is consumed in idle state. We also focus on the number of retransmissions at the MAC layer and at the TCP layer. At the MAC level, we measure the aggregate number of *RTS retransmissions* due to the expiration of the timeout for the reception of the CTS and the aggregate number of *Packet retransmissions* due to the timeout for the reception of the corresponding ACK message. At the transport level, we measure the aggregate number of *message retransmissions* and the aggregate number of *message fast-retransmissions*. We also evaluate the *Average Download Rate* metric derived from the content size and the TTD distribution.

IV. Simulation results

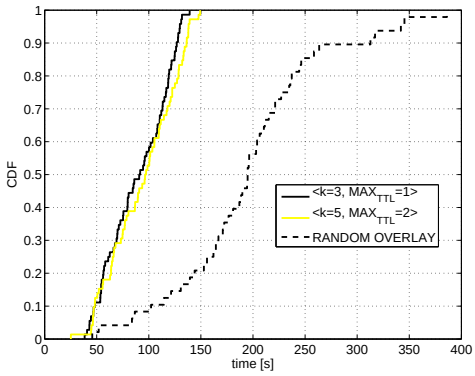
A. Impact of the overlay structure

In this section we examine the impact of the overlay structure on the performance of our mechanism in terms of the TTD metric. We vary the target size k (which is the same for every node) and the MAX_{TTL} and study their influence on two representative scenarios (topology 1 and 2).

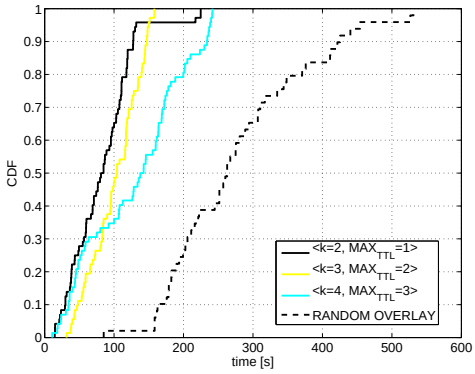
Our goal is to evaluate empirically the parameters that yield lower TTD values. In general, the problem we address depends on the density of the network and on the popularity of the content. We simplify the problem by focusing only on the dependence on the network density, as we set the content popularity to be 100%. For a fixed target size k, the choice of MAX_{TTL} depends on the physical node degree: for a dense network the hop distance to search for overlay neighbors can be smaller than for a sparse network. Note that the choice of k also has an impact on the the TTD metric, as shown in [24].

For a dense network (see Figure 1), we study the cases $< k = 3, MAX_{TTL} = 1 >$ and $< k = 5, MAX_{TTL} = 2 >$. For a sparse network (see Figure 2) we use the following configurations: $< k = 2, MAX_{TTL} = 1 >$, $< k = 3, MAX_{TTL} = 2 >$ and $< k = 4, MAX_{TTL} = 3 >$. Figures 3(a) and 3(b), show the CDF of the TTD for the two reference networks.

The results in Figure 3(a) show that for a dense network, increasing the target size k and expanding the search horizon MAX_{TTL} has a negligible effect on the TTD cumulative distribution. This is mainly due to interference and collision issues at the MAC level that we will analyze in more detail in Section IV-C. As opposed to what has been observed for



(a) Results for topology 1



(b) Results for topology 2

Fig. 3. Impact of the overlay structure: distribution of the time to download the content for different network topologies. With our technique an increase of the number of outgoing connections (OC) can improve performances at the cost of expanding the search ring. System performance severely degrade with a randomized approach to build the distribution overlay.

CCD mechanisms used in a wireline context such as the Internet (see for example [24]) the benefits that derive from a "rich" neighbor set (both in terms of peers and piece diversity) are partially neutralized by the shared nature of the wireless medium.

In contrast, Figure 3(b) shows that the simple heuristic of limiting the $MAX_{TTL} = 1$ might be sub-optimal for sparse networks. In this case, selecting only physical neighbors to take part in the overlay might result in some nodes (see for example node 4 and 6 in Figure 2) to be dependent on one peer only to obtain the content. This explains the small fraction of nodes with TTD values greater then 200 seconds. However, extending the search perimeter using $MAX_{TTL} = 3$ results in a performance degradation due to longer underlay routes to exchange pieces of the content. In Figure 3(b) we show that the configuration $< k = 3, MAX_{TTL} = 2 >$ yields a TTD distribution equivalent to the one obtained in Figure 3(a), with a maximum time to distribute the content of approximatively 150 seconds.

For the sake of completeness, we compare our technique with a centralized approach equivalent to the *tracker* component in BitTorrent [1]. We simulate the presence of a central

entity to bootstrap the distribution overlay in a random fashion: each peer is provided with $k = 4$ neighbors to connect to, chosen at random among all the nodes of the network. Figures 3(a) and 3(b) show that traffic locality determines to a large extent the performance of our CCD mechanism. Both the median and the variance of the TTD increase when a randomized approach is used.

We also studied the impact of the position of the initial seed in a dense network. Figure 4 shows that a central or peripheral position have a little impact on the performance of our solution. Indeed, because of our particular distribution overlay, the content propagates along routes with a bounded hop count.

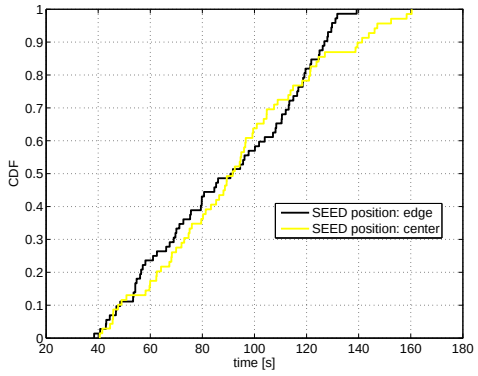


Fig. 4. Impact of the seed position on the CDF of the time to download the content for the topology in Figure 1. The impact of a central or peripheral position of the content source does not severely impact the solution proposed in this paper.

B. Detailed protocol behavior

In this section we explore the behavior of the proposed CCD mechanism in more detail. We chose the parameters of the overlay construction technique such that $k = 3, MAX_{TTL} = 2$. We only present results for the topology in Figure 1, as similar results hold for the latter topology.

In Figures 5, 6 and 7 we show the piece propagation, the aggregate upload capacity of the system and the number of parallel downloads of our system.

Figure 5 illustrates a relatively high propagation speed for the first piece of the content: in less than 30 seconds all leechers get their first piece. However, the initial thrust is not sustained throughout the content distribution process, as the dispersion in the reception time of the last piece indicates. In Section IV-C we show that this decrease in performance is mainly due to MAC level retransmissions.

Figure 6 shows that the average service capacity of the system is approximatively 850 KBps. Pieces of the content are injected in the system at a relatively high pace.

In Figure 7 we show a metric related to the spatial reuse of the network. The localized approach used to build the distribution overlay allows multiple communications to take place at the same time, as shown by the high number of parallel downloads. The drop in the number of parallel downloads corresponds to the system reaching a full regime (*i.e.* all peers are able to trade pieces). When all peers have data to transmit interference and MAC-level collisions are substantial. Note, however, that the decrease in number of parallel downloads is also due to an increasing number of peers that completed their download (see Figures 5 and 7).

C. A baseline scenario

In this section we further analyze the performance of our CCD scheme using measures taken at different levels of the protocol stack. Note that our goal here is not to compare the performance of our approach with alternative applications for content distribution. A fair performance comparison should be done for example with application layer multicast [10]. In contrast, we investigate on the characteristics of our scheme using a baseline scenario as a lower-bound reference. We use as a basis the performance of a naive client-server (FTP) approach: an FTP server runs on the source and all the other nodes in the network fetch the content using an FTP client.

Figure 8 and 10 show respectively the CDF of the TTD for networks in Figures 1 and 2 for the CCD and the FTP schemes. We observe that for both network types the median of the TTD is smaller for the CCD case (roughly 100 seconds as compared to 180 seconds). Similarly, the performance of our scheme is above the baseline scenario when considering the variance and the maximum of the TTD. We further remark that the FTP solution is more sensitive to the network density than our solution. Indeed, the maximum time for the content to be distributed in a sparse network doubles as compared to a dense network. The literature is rich of mechanisms, such as cooperative caching [22], [26], used to mitigate these problems although they don't fall within the scope of our work.

The features of our CCD scheme appear evident if we measure the amount of energy consumed by the nodes. Figure 9 and 11 show that in the baseline scenario nodes consume *an order of magnitude* more energy than in our solution. Moreover, we observe that in our solution the seed does not consume more energy than any other peer, and that peers consume roughly the same amount of energy.

To explain the difference in energetic consumption between the CCD and the baseline case, we focus on the topology in Figure 1. In Figure 12 we show the average number of retransmissions at MAC and transport layer, as explained in Section III-B. While the transport layer is affected similarly in both schemes (right plot in Figure 12), the number of retransmissions at the MAC layer is the main factor that differentiate the CCD from the baseline scenario. The loss of MAC-level control and data packets triggers the exponential backoff mechanism typical of the 802.11 protocol, which leads a node to wait an increasingly larger amount of time before (re)transmitting a packet. Hence the poor performance and the higher energy consumption in the FTP case. Note that our scheme is also affected by MAC-level retransmissions: however, our approach in building the CCD overlay drastically reduces the number of unnecessary retransmissions due to interference and collisions.

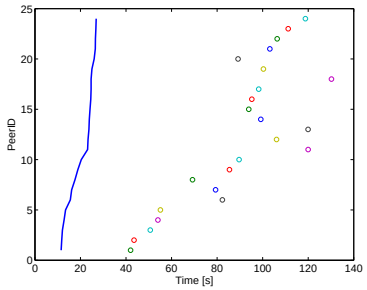


Fig. 5. First and last piece download progress for the topology in Figure 1. PeersID are ordered based on the first piece reception time

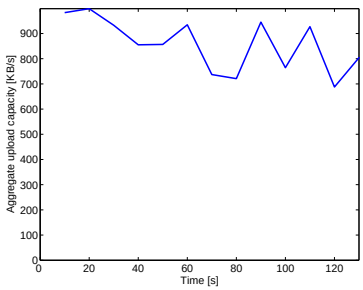


Fig. 6. Aggregate upload capacity of the system in a 10 sec. time interval for the topology in Figure 1.

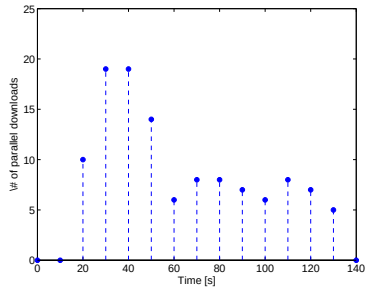


Fig. 7. Number of parallel downloads in a 10 sec. time interval for the topology in Figure 1.

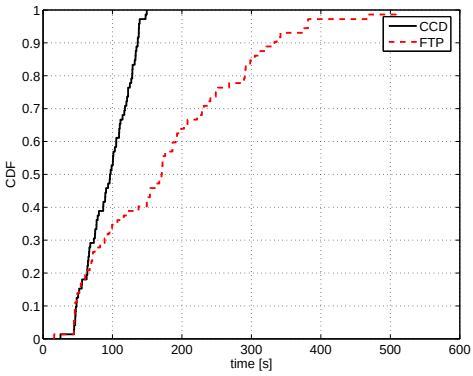


Fig. 8. Cumulative distribution of the time to download the content for our CCD scheme and for the FTP mechanism for the topology in Figure 1. The CCD case outperforms the FTP case for both the median and the variance of the TTD distribution.

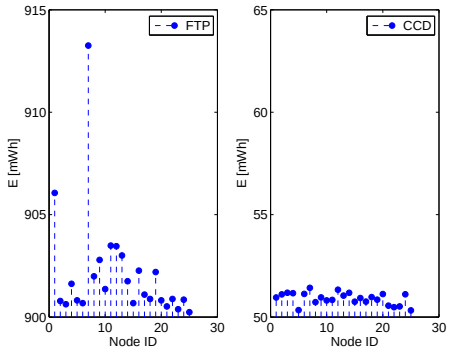


Fig. 9. Energy consumption comparison between CCD and FTP case for the topology in Figure 1. The CCD solution remarkably consumes less energy (an order of magnitude) and has the property of equalizing the costs associated to the content distribution among all peers.

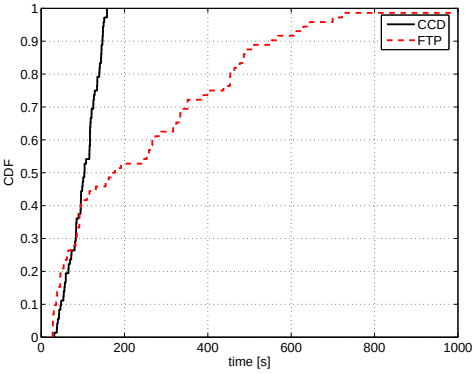


Fig. 10. Cumulative distribution of the time to download the content for the CCD and the FTP case for the topology in Figure 2. The maximum time to distribute the content for the FTP case is higher as compared to Figure 8 because of the underlying network topology, while the CCD scheme is not highly impacted.

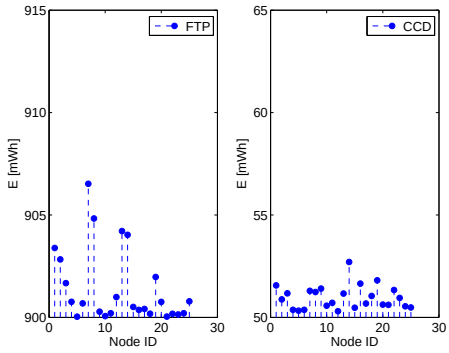


Fig. 11. Energy consumption comparison between CCD and FTP case for the topology in Figure 2. The difference in energy consumption is remarkable for the CCD and FTP cases.

D. Scalability properties

In this Section we study the scalability properties of our CCD mechanism. In our experiments we follow the *extended network model* [7] whereby we maintain the density of the nodes in the network constant, while increasing both the number of nodes and the (square) area on which nodes are deployed. We scale the network up to 100 nodes and use a control script to rule out partitioned networks before executing the simulation.

Results are presented in Figure 13, where we show the average download throughput of a peer, *i.e.* the *per-peer capacity*.

With the sake of fitting our experimental values to an analytical expression we report a scaled version of the per-node capacity law $\Theta(1/\sqrt{n \log n})$ as introduced by Gupta *et. al.* in [13]. Although the scaling law obtained in [13]

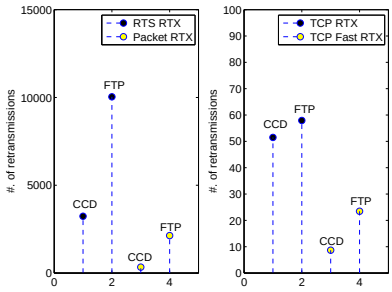


Fig. 12. Average number of retransmissions at the MAC and Transport Layers for the topology in Figure 1. The impact of MAC retransmissions due to interference and collisions is remarkable as compared to retransmissions at the transport layer.

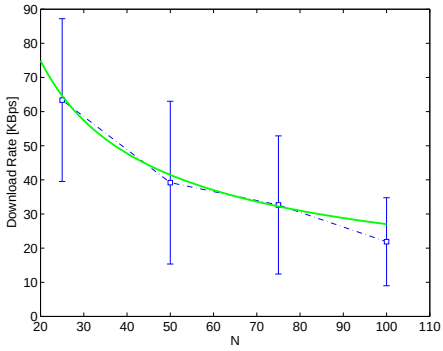


Fig. 13. Per node capacity scaling under the extended network model. The network size has a severe impact on the per node capacity. In the Figure we plot a scaled version of the $1/sqrt(nlogn)$ law for sake of comparison with our experimental results.

assumed a random network deployed on a unit disk area (we deploy our nodes on a square area), optimal scheduling of packet transmission and random source-destination pairs the experimental and the analytical curves loosely match and differ only by a constant factor. Our scheme scales no worse than what has been predicted in [13]. Note that our results hold for a realistic packet scheduling; furthermore, using random source-destination pairs (*i.e.* a random overlay) would have yield worse performance, as shown in Section IV-A; lastly, current analytical models neglect the effects of a transport protocol such as TCP on the scaling law of the network.

V. Protocol overhead

We present now the per-peer overhead of our protocol. At the application layer, the protocol overhead is due to control messages used by the piece scheduling algorithm to infer piece statistical distribution. In our computation we assume that: control messages are not lost, which is reasonable since we use TCP for their transmission; there are no peers joining or leaving the overlay network; and that the suppression technique explained in section II-B is not used. Then, every peer in the system exchanges one control message (of $1byte$) with every other peer in its overlay neighborhood for every piece of the content. Thus the per-peer overhead is $(k-1)*(k-1)*c\,bytes$, where k is the size of the neighbor set of each peer and c is the number of pieces the content. We also need to count the handshake messages exchanged by peers to establish overlay connections, which are $c\,bit$ long (see Section II-B).

For example, in our simulations we have $k = 3$, $c = 20$: the per-peer overhead for a $5MB$ file is $10bytes$ for the handshake messages and $80bytes$ for subsequent control messages, which constitutes a negligible overhead.

VI. Related work

Peer-to-peer applications for mobile ad hoc networks have recently gained a lot of attention in the research community. Several works [6], [8], [17], [23] have studied efficient lookup mechanisms with the goal of overcoming the issues due to node mobility. Some works, such as [6], [17] focus on DHT-based techniques, enriched with positioning information and compare them with alternative approaches based on optimized flooding. In contrast, [8], [23] propose search techniques based on optimized flooding. These works have been carried out with the ultimate goal of mitigating the impact of node mobility on the query response time and on content availability.

Other works [10], [12] focus on application layer multicast for mobile ad hoc networks. These works study the effects of mobility on routing efficiency and on the construction of the overlay mutlicast tree. In principle, the construction of multicast overlays is similar to the problem discussed in this paper, although the resulting logical structure is more rigid. Hence, studying the impact of mobility on overlay maintenance is fundamental. Furthermore, a performance analysis of the content distribution has not been the main focus of these works.

Works closely related to ours are [14], [19]–[21]. In [14], Klemm *et. al.* propose a complete p2p system, ORION that features a search and an original data transfer mechanism. In ORION, the transfer protocol schedules the transmission of parts of the content based on node mobility and implements a simple retransmission mechanism to recover from failures. Peer cooperation is not taken into account. The simulation-based performance evaluation of ORION targets user satisfaction in terms of the percentage of successful data transfers rather than effective data transfer rates; moreover, the content is assumed to be already replicated on some specific nodes.

In [19], the authors present a content distribution scheme for vehicular networks and validate it using simulations, emphasizing the effects of mobility. The design is inspired by the BitTorrent protocol: they propose a random overlay construction mechanism and modify the original piece scheduling algorithm to take into account both statistical information (rarity) and distance (in hops) from the requesting peer. However, the authors do not evaluate the impact of such a choice on the diversity of piece distribution in the system. In contrast, recent works [16], [18] show that piece diversity achieved by the scheduling algorithm in BitTorrent (that we also use in our work) plays an important role on the efficiency of the system.

In [20], [21] the authors adapt the BitTorrent protocol to mobile ad hoc networks and study the effects of mobility on the performance of their scheme. However, they use a random piece scheduling algorithm and an overlay construction mechanism tightly coupled with an ad hoc routing protocol through cross-layer optimization. They do not take into account peer cooperation and introduce the notion of proxy peers, that is they replicate the original content at multiple locations to help with the distribution process. The validation of the scheme is limited to very unpopular contents distributed to a small (three) set of destinations. To the best of our knowledge an accurate performance evaluation of cooperative content distribution and its salient features on ad hoc networks is not available in the literature. We use algorithms that exhibit exceptional performance in the Internet [16] and adapt them to a context wherein interference and multi-hop routing invalidate the typical assumption that the bottleneck for the capacity of peers to serve a content is due to a limited access link capacity, whereas the capacity of the core network is traditionally assumed to be unconstrained. We study these issues in our simulations and show that our scheme performs remarkably well under different network topologies. Moreover, as opposed to [20], our mechanism is routing independent.

VII. Conclusion and future work

In this paper we study the performance achieved by our cooperative content distribution scheme in a static multi-hop network. Our scheme features a decentralized mechanism to build a wireless overlay, an algorithm to trade pieces of the content among peers and an algorithm to foster peer collaboration. Rather than on mobility, we focus on the performance limitations imposed by the underlying multi-hop network and the shared nature of the medium used by wireless nodes to communicate. We mitigate these problems by taking into account traffic locality as a key to build the wireless overlay. With our solution the cost associated to the content distribution is evenly shared among the peers and cooperation is fostered.

Our simulation results show that the performance in terms of the total time to distribute the content of our mechanism are reasonably good for scenarios characterized by different network topologies. We show that the energy consumption is uniformly shared by peers and that our scheme partially mitigates the effects of MAC-level retransmissions due to interference and collisions. We conclude our analysis showing that the capacity scaling of our solution decreases as the one predicted by analytical models for an ideal system and that the protocol overhead is low.

In our future research we will present performance results for the mobile case and focus on new scenarios whereby we will vary the content popularity, the peer departure rates and the peer cooperation level. We will also explore the potential benefits of using network coding as a replacement of our piece scheduling algorithm.

Acknowledgments

This work has been partially funded by the Integrated Project CASCADAS (FET Proactive Initiative, IST-2004-2.3.4 Situated and Autonomic Communications) within the 6th IST Framework Program.

References

[1] Bittorrent protocol. http://wiki.theory.org/BitTorrentSpecification.
[2] Optimized link state routing protocol: Internet draft. http://hipercom.inria.fr/olsr/draft-ietf-manet-olsr-11.txt.
[3] QualNet simulation suite. http://scalable-networks.com.
[4] A. Bharambe, C. Herley, and N. Venkata. Analysing and Improving BitTorrent Performance. In *Proc. of IEEE INFOCOM*, Barcelona, Spain, 2005.
[5] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, USA, 2003.
[6] G. Ding and 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] O. Dousse and P. Thiran. Connectivity vs capacity in dense ad hoc networks. In *Proc. of IEEE INFOCOM*, Hong Kong, China, 2004.
[8] A. Duran and C. Shen. Mobile ad hoc P2P file sharing. In *Proc. of IEEE WCNC*, Atlanta, GE, USA, 2004.
[9] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. of IEEE INFOCOM*, Anchorage AK, USA, 2001.
[10] M. Ge, S. V. Krishnamurthy, and M. Faloutsos. Application versus Network Layer Multicasting in Ad Hoc Networks: The ALMA Routing Protocol. *Ad Hoc Networks Journal (Elsevier)*, 4(2):283–300, 2006.
[11] M. Gerla, K. Tang, and R. Bagrodia. TCP Performance in Wireless Multi-hop Networks. In *Proc. of IEEE WMCSA*, Washington, DC, USA, 1999.
[12] C. Gui and P. Mohapatra. Efficient Overlay Multicast for Mobile Ad Hoc Networks. In *Proc. of IEEE WCNC*, New Orleans, LA, USA, 2003.
[13] P. Gupta and P. R. Kumar. The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404, 2000.
[14] A. Klemm, C. Lindemann, and O. P. Waldhorst. A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks. In *Proc. of IEEE VTC-Fall*, Orlando, Florida, USA, 2003.
[15] J. Lee, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of Ad Hoc wireless networks. In *Proc. of IEEE/ACM MOBICOM*, Rome, Italy, 2001.
[16] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *Proc. ACM SIGCOMM/USENIX IMC*, 2006.
[17] M. Li, W-C. Lee, and A. Sivasubramaniam. Efficient Peer-to-Peer Information Sharing over Mobile Ad Hoc Networks. In *Proc. of MobEA*, 2002.
[18] P. Michiardi and G. Urvoy-Keller. Cooperative Content Distribution for Ad hoc Networks: the static case. Technical Report RR-06-172, Institut Eurecom, 2006.
[19] A. Nandan, S. Das, G. Pau, and M. Gerla. Co-operative downloading in Vehicular Ad-hoc Wireless Networks. In *Proc. of IEEE WONS*, Washington, DC, USA, 2005.
[20] S. Rajagopalan and C-C. Shen. A Cross-Layer, Decentralized BitTorrent for Mobile Ad hoc Networks. In *Proc. of ACM/IEEE MOBIQUITOUS*, San Jose, CA, USA, 2006.
[21] S. Rajagopalan and C-C. Shen. Is BitTorrent for Mobile Ad hoc Networks Faster than Traditional P2P? In *Proc. of ACM MOBISHARE*, Los Angeles, CA, USA, 2006.
[22] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *Proc. MDM*, 2003.
[23] M. Y. Sung, J. H. Lee, and Y. J. Heo. Towards reliable peer-to-peer data sharing over mobile ad hoc networks. In *Proc. of IEEE VTC-Spring*, Stockholm, Sweden, 2005.
[24] G. Urvoy-Keller and P. Michiardi. Impact of Inner Parameters and Overlay Structure on the Performance of BitTorrent. In *Proc. of the 9th IEEE Global Internet Symposium Workshop, In Conjunction with IEEE INFOCOM*, Barcelona, Spain, 2006.
[25] X. Yang and G. de Veciana. Service Capacity in Peer-to-Peer Networks. In *Proc. of IEEE INFOCOM*, Hong Kong, China, 2004.
[26] L. Yin and G. Cao. Supporting co-operative caching in ad hoc networks. In *Proc. of IEEE INFOCOM*, 2004.