Efficient Peer-to-peer File Sharing using Network Coding in MANET

Uichin Lee, Joon-Sang Park, Seung-Hoon Lee, Won W. Ro, Giovanni Pau, Mario Gerla
{uclee,shlee,gpau,gerla}@cs.ucla.edu, jsp@hongik.ac.kr, wro@yonsei.ac.kr

***Abstract*—Mobile peer-to-peer systems have recently got in the limelight of the research community that is striving to build efficient and effective mobile content addressable networks. Along this line of research, we propose a new peer-to-peer (P2P) file sharing protocol suited to mobile ad hoc networks (MANET). The main ingredients of our protocol are network coding and mobility assisted data propagation, i.e., single-hop communication. We argue that network coding in combination with single-hop communication allows P2P file sharing systems in MANET to operate in a more efficient manner and helps the systems to deal with typical MANET issues such as dynamic topology and intermittent connectivity as well as various other issues that have been disregarded in previous MANET P2P researches such as addressing, node/user density, non-cooperativeness, and unreliable channel. Via simulation, we show that our P2P protocol based on network coding and single-hop communication allows shorter file downloading delays compared to an existing MANET P2P protocol.**

***Index Terms*—File swarming, mobile ad hoc networks, mobility-assisted, network coding, peer-to-peer, random linear code.**

I. Introduction

Peer-to-peer (P2P) file sharing in mobile ad hoc networks has continuously gained popularity due to its strong adaptability in many practical applications. However, using existing P2P file sharing systems (such as Gnutella and BitTorrent) on mobile ad hoc networks (MANET) inherently contains some limitations since they are mainly developed for the wired networks. Recently, several P2P schemes targeting MANET have been proposed as in MANET-optimized versions of existing P2P as well as clean-slate designs. Most of the recently proposed MANET P2P protocols attempt to address the problems caused by dynamic topology through cross-layer optimization since MANET is characterized by highly dynamic topology; however, P2P protocols are encumbered with various other characteristics of MANET.

First problem is caused by the fact that the wireless channel is error prone. If a protocol is designed without considering potential errors, the performance of the protocol in real deployment will be seriously degraded. For example, TCP connections usually die out in multihop networks with lossy channel but most P2P protocols simply assume that TCP offers reasonable bandwidth. Secondly, number of users and user density should also be considered. In a file sharing scenario, the total number of users can scale up to tens of thousands of nodes, and theoretically, all of the nodes can be users running P2P protocols. Even with any cross-layer optimization, no conventional MANET routing protocols is expected to support such big networks. Possible non-cooperative nodes are another concern. Most MANET protocols are designed based on the assumption of node cooperativeness. Multihop routes can only be established when there are nodes willing to serve as relays for the sake of data sender. In a MANET built/maintained/owned by a single entity, such as a military tactical network or wireless mesh network, nodes can easily be forced to cooperate to achieve a common goal (e.g., providing a communication infrastructure.) But in other types of MANET such as vehicular ad hoc networks (VANET) consisting of cars, trucks, or any other types of vehicles on the road, it is very likely that nodes are operated by different entities for their own good and thus it may not possible to force every node to cooperate each other. Lastly, IP addressing is non-trivial in large scale MANET. It is not clear how each node will be assigned an IP address in a large scale MANET such as VANET.

In this paper, we investigate the problem of running BitTorrent type P2P file sharing systems, i.e., file swarming protocols, in MANET. To remedy the issues identified above, we take a holistic approach. Put another way, instead of solving each issue separately as an independent problem, we design an entirely new protocol to address all these problems at once. As we can see, the use of existing MANET routing protocols in P2P file swarming give rise to most of the issues. In our design, we resort to single-hop communication. Multihop routes are never used and thus are not required to be maintained explicitly by any layer in the protocol stack. Rather, multihop communication is implicit. We restrict logical peers, i.e., nodes exchanging file pieces, to physical neighbors, yet data is propagated through the (overlay) network of peers of common interest, which is the basic concept of operation of P2P file sharing systems. The main problem of restricting logical peers to physical neighbors in MANET, however, is connectivity amongst peers. It might be difficult for a node to find peers of common interest. Even though some peers are found, there is no guarantee that those peers possess useful data. The main ingredients of our design are network coding and mobility assisted data propagation (e.g. [2]). The two techniques allow our design to maintain enough connectivity among peers with low overhead such that users can download files in less time than existing protocols.

By network coding, we refer to the notion of performing coding operations on the contents of packets throughout a network. This notion is generally attributed to Ahlswede et

This work in part presented in ACM MobiShare 2006, the 1st International Workshop on Decentralized Resource Sharing in Mobile Networks

al. [1], who showed the utility of the network coding for multicast. The work of Ahlswede et al. was followed by other work by Koetter and Médard [12] that showed that codes with a simple, linear structure were sufficient to achieve the capacity of multicast connections in lossless, wireline networks. This result was augmented by Ho et al. [9], who showed that a random construction of the linear codes was sufficient. The utility of such random linear code for wired P2P file sharing systems was soon realized in [8]. Network coding improves the performance of P2P file swarming systems since it mitigates the block transfer scheduling or piece selection problem especially when only local information is given and nodes dynamically join/depart. It helps increase the number of distinct pieces available in the network via coding, thus providing a higher chance for peers to pull useful pieces [4], [8]. Recently, implementation and performance issues in network coding based P2P have been investigated in [23], [22], [14]. Our contribution in this lineage is that we show the utility of random linear code for P2P file swarming protocols "in MANET." Our work is inspired by [8] in which the performance advantage of using random linear code in "wired" P2P systems is investigated. Different from [8], we show that network coding helps exploit unique opportunities offered in the MANET P2P environment, the broadcast nature of wireless medium and node mobility. Our file swarming protocol based on network coding and mobility assisted data propagation, i.e., single-hop communication, shows less download delay than an existing MANET file swarming protocol.

The rest of this paper is organized as follows. Section II illustrates our network coding based file swarming protocol and we evaluate the protocol through simulation in Section III. Section IV presents the related work and, finally, Section V concludes this paper.

II. Network Coding based File Swarming Protocol

In this section, we describe our file swarming protocol for MANET named CodeTorrent. To start, we define a seed node to be a node which possesses a complete file and has intension to share the file. The seed node announces the availability of the file via one-hop broadcast of the *description* of the file. Similar to the torrent file in BitTorrent protocol, a description contains, for example, identification number, file name, file size, number of pieces, etc. We simply assume that each file can be uniquely identified with an identification number (*fileid*) during the time period in which every node interested in the file completes its downloading.

At the seed node, a file F is divided into n pieces $\mathbf{p}_1$, $\mathbf{p}_2$, ..., $\mathbf{p}_n$. In our protocol, nodes exchange *coded frames* instead of file pieces. We define a coded frame $\mathbf{c}$ to be a linear combination of file piece $\mathbf{p}_k$'s. That is, $\mathbf{c} = \sum_{k=1}^{n} e_k \mathbf{p}_k$ where e_k is a certain element in a certain finite field $\mathbb{F}$ over which every arithmetic operation is. File piece $\mathbf{p}$'s and coded frame $\mathbf{c}$'s are also regarded as vectors over $\mathbb{F}$. Whenever the seed node is requested to exchange a coded frame, the node transmits a newly generated a coded frame $\mathbf{c}$ and when generating $\mathbf{c}$, each e_k is drawn randomly from $\mathbb{F}$, hence the name of random linear coding (RLC). In the header of a coded frame, the *encoding vector* $\mathbf{e} = [e_1 \ ... \ e_n]$ is stored for the purpose of later *decoding* [5]. Throughout this paper, we use lowercase boldface letters to denote vectors, frames, or packets, uppercase letters to denote matrices, italics to denote variables.

A node learns of a file from receiving the file's description transmitted from neighbors. If the node finds the file interesting, it broadcast a request containing *fileid* of the file and the transaction ID. Upon receiving such a request, the node responds with a newly generated coded frame. A node may receive a request even though it does not possess the complete file, i.e., it is not a seed node. If it possesses any coded frame of the requested file, it should respond to the request. The reply, i.e., a coded frame, is accompanied by the corresponding transaction ID. A node keeps requesting neighbors to send coded frames until it collects n coded frames carrying encoding vectors that are linearly independent of each other. Simultaneous requests and replies can be distinguished and matched by their transaction IDs. Every node announces the availability of any coded frame it possesses. Not only the seed node of a file also every node which possesses any coded frame of the file and willing to share them periodically broadcasts to its 1-hop neighbors the description of the file. If a node has multiple files to share, multiple descriptions are packed into the least number of packets that can carry all of them and then transmitted.

Whenever requested, every node, not just the seed node, creates on-the-fly and transmits a new coded frame. To create a "new" code frames on a non-seed node which is defined to be a node possessing a number of coded frame less than n, the coded frames stored in local memory are used. A non-seed node generates a random linear combination of coded frames available in local memory and transmits it. Since the frames in local memory are already coded ones, the "re-encoded" frame to be transmitted $\acute{\mathbf{c}} = \sum_{k=1}^{rnk} \acute{e}_k \mathbf{c}_k$ is tagged with the encoding vector $\acute{\mathbf{e}} = \sum_{k=1}^{rnk} \acute{e}_k \mathbf{e}_k$ where $\mathbf{c}_k$ and $\mathbf{e}_k$ is a coded frame in local memory and the encoding vector prefixed to $\mathbf{c}_k$ respectively. rnk is the number of $\mathbf{c}_k$'s found in local memory. When encoding, each $\acute{e}_k$ is drawn uniformly from $\mathbb{F}$.

To recover n file pieces $\mathbf{p}_1$, $\mathbf{p}_2$, ..., $\mathbf{p}_n$, a node must collect more than n coded frames carrying encoding vectors that are linearly independent of each other. Let $\mathbf{c}_k$ be a coded frame, $\mathbf{e}_k$ be the encoding vector prefixed to $\mathbf{c}_k$, and $\mathbf{p}_k$ be a file piece to be decoded and recovered where k = 1, ..., n. Further, let $\mathbf{E}^{\mathrm{T}} = [\mathbf{e}_1^{\mathrm{T}} \ ... \ \mathbf{e}_n^{\mathrm{T}}]$, $\mathbf{C}^{\mathrm{T}} = [\mathbf{c}_1^{\mathrm{T}} \ ... \ \mathbf{c}_n^{\mathrm{T}}]$, and $\mathbf{P}^{\mathrm{T}} = [\mathbf{p}_1^{\mathrm{T}} \ ... \ \mathbf{p}_n^{\mathrm{T}}]$ where superscript T denotes the transpose operation. The original file pieces are obtained from $\mathbf{P} = \mathbf{E}^{-1}\mathbf{C}$. Note that all $\mathbf{e}_k$'s must be linearly independent to be able to invert $\mathbf{E}$.

A request of coded frames may be accompanied by the *nullspace vector* which is a vector in the nullspace spanned by all encoding vectors of the frames stored in the local memory of the requesting node. On reception of such a request, a node transmits a coded frame only if there is in its local memory a frame with the encoding vector that is not orthogonal to the nullspace vector received with the request.

Every node promiscuously listens to packets, i.e., a node receives a specific packet even the node is not the designated receiver, so that it can use them if possible. A node always

overhears the packets carrying coded frames and treats the overheard ones as the coded frames transmitted specifically to the node and if an overheard coded frame is linearly independent of the coded frames in local memory, the node stores it in local memory.

Since every transmission is MAC/link layer broadcasting, a small random amount of wait time before each transmission called broadcast jitter is applied to reduce collisions. Without broadcast jitter, MAC/link layer broadcasting suffers severely from the hidden terminal problem.

If shared files are large ones, there are additional issues. One of them is the size of encoding vector. The size of encoding vector is proportional to the file size divided by coded packet size. Assuming GF(2^8), if the file size is 1GB and the packet size is 1KB, then the size of encoding vector is 1MB which is to large to fit in the header of coded frames. One solution to this issue is to use the concept of "generation" [5] as follows. The original file is divided into m generations. Each generation i has g pieces (we call g the *generation size*) and the *piece size* is fixed to b KB: i.e., $\mathbf{p}_{i,1}$, $\mathbf{p}_{i,2}, \cdots, \mathbf{p}_{i,g}$ for $i = 1, \cdots, m$. For each generation i, the seed node creates a coded piece via *random linear coding* over all the pieces in the same generation: $\sum_{k=1}^{g} c_k \mathbf{p}_{i,k}$ where c_k is randomly drawn over $\mathbb{F}$. Each intermediate node similarly generates a coded piece by combining all the coded pieces collected so far for that generation and only keeps linearly independent coded pieces. Each coded piece is marked with the *generation number*, and coded pieces belonging to the *same* generation are used for encoding. For a given generation, after collecting g coded pieces that are linearly independent of each other, a node can recover the original data by simply solving a set of linear equations. This process repeats until the node collects all m generations.

III. Evaluation

In this section we evaluate and compare the performance of CodeTorrent to CarTorrent [16], an earliest file swarming protocol for MANET, through simulations using Qualnet [20]. In the simulations, we use IEEE 802.11b PHY/MAC with 2Mbps bandwidth and Random Waypoint mobility model. A fraction of nodes (denoted as popularity) in the network is interested in downloading the same file. There is a special type of node called AP which possesses the complete file at the beginning of each simulation. Three static APs are randomly positioned on the 2400m x 2400m field. In CarTorrent, we use UDP to transfer data packets. As the underlying routing protocol, we use AODV [19]. We limit the scope of the gossip packets (which tell who possesses what) to 3 hops as proposed in the original design [16]. We also limit the TTL value of RREQ in AODV to 3 hops (same as gossip packets.) Each node initiates a piece downloading either periodically (i.e., every 0.5 second) or upon receiving a new gossip packet. Successful downloading of a piece will also initiate downloading of another piece. Piece availability gossiping is carried out for every 5 seconds. We use a probabilistic gossiping: uninterested and interested nodes forwarded the gossiping packets with probability 0.1 and 0.8 respectively. Similarly,

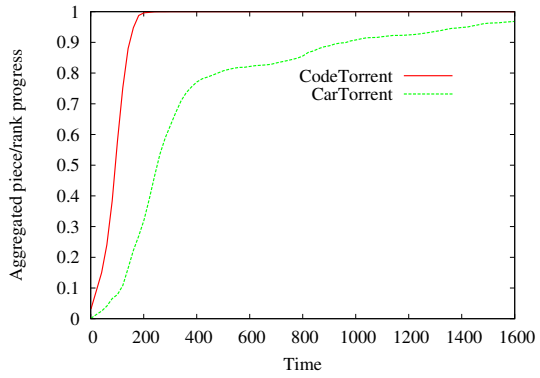

Fig. 1. Aggregated downloading progress (200 nodes moving at maximum speed of 20 m/s with 40% popularity)

CodeTorrent uses UDP to transfer packets to its neighbors. CodeTorrent does not use any underlying routing protocol, because it only relies on the single hop unicast. Overhearing is allowed in CodeTorrent (i.e., every node promiscuously listen to the wireless channel and receive packets regardless of their destinations.) and GF(2^8) is used for coding.

Unless otherwise specified, the sharing file is 1MB sized and a single generation constitutes the file. The block or piece size is 4KB and thus there are 250 pieces total in the generation. In the CodeTorrent case, a peer must acquire 250 linearly independent coded pieces to decode the file. A coded piece with encoding vector prefixed is transferred using multiple 1KB packets. Since the size of the encoding vector is 250B, about 6% of each packet is the overhead paid for the encoding vector.

We define the download "delay" to be the elapsed time for a node to collect all the pieces constituting the file for CarTorrent or a enough number (same as the generation size) of linearly independent coded pieces from which the original file can be recovered for CodeTorrent. The given metric is evaluated with various configurations, i.e., as a function of node density, maximum node speed, and fraction of interested nodes.

A. Comparison of Download Delay

First, we contrast the download delay of CodeTorrent to that of CarTorrent in a specific setting to show the performance benefit of CodeTorrent over CarTorrent.

The aggregated downloading progress (cumulative histogram with slot size of 20 seconds) is shown in Figure 1. For each time slot (x-axis), the figure plots the average fraction of pieces collected by 80 nodes for CarTorrent and the averaged fraction of linearly independent coded pieces collected by 80 nodes for CodeTorrent. The figure clearly indicates that CodeTorrent significantly expedites the overall progress compared to CarTorrent.

Figure 2 shows the histogram of download delays for both protocols with a slot size of 20 seconds. In CodeTorrent, nodes collectively help each other to distribute coded pieces using network coding (i.e., algebraic mixing) and overhearing. At the second time slot, i.e., [20,40), we see that around 5 nodes

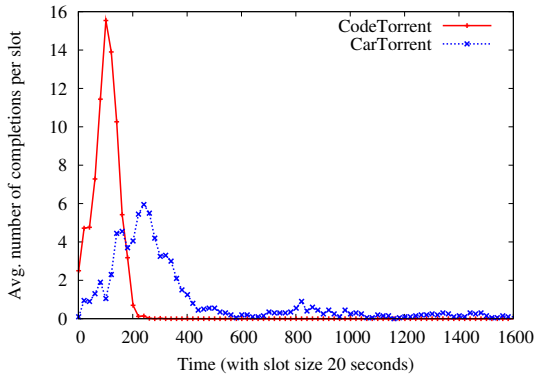

Fig. 2. Histogram of download delays

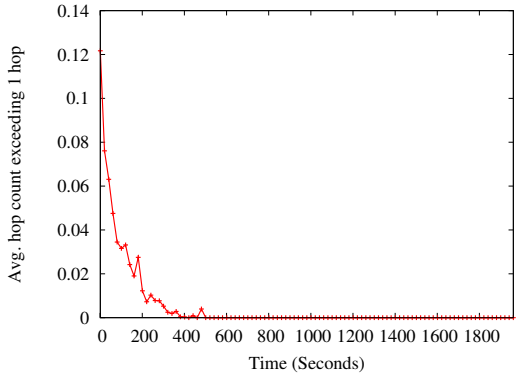

Fig. 3. Average hop count histogram for multi-hop pulling in CarTorrent

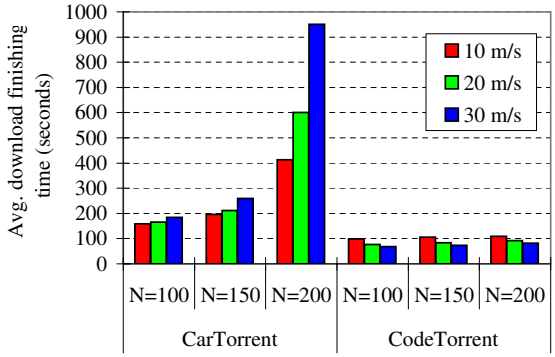

Fig. 4. Impact of mobility on average download delay

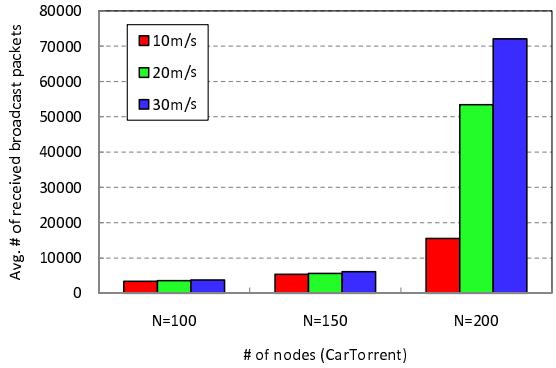

Fig. 5. Overhead in CarTorrent

become seeds (a node becomes a seed when it completes downloading of the shared file), which is followed by a burst of about 15 new seeds in a few next slots. As the number of seeds increases in the network, the usefulness of random coded packet increases and thus, this further shortens downloading time.

On the other hand, CarTorrent does not show such burst births of seeds, but seeds are born rather gradually. This is mainly due to the competition among nodes to secure downloading bandwidth. For example, after receiving a gossip packet from nodes or APs, in the worst case, 80 interested nodes start requesting pieces all at the same time. Since overhearing is not assumes, a number of nodes inevitably contend for the limited bandwidth. This crowd effect causes severe channel contention, thus resulting in performance degradation as shown in Figure 2. Moreover, some peers that are not one-hop physical neighbors to each other necessitate multi-hop communication which aggravates the channel contention. In Figure 2, it shows that the first download completion happened at the second slot and the maximum birth rate of seeds[1] was always less than 5. To show the behavior of multi-hop pulling, in Figure 3 we plot the histogram of average hop count exceeding 1 hop with a slot size of 20 seconds. The figure clearly states that since the availability of a random piece increases as time passes, the average hop count gradually decreases. Multi-hop pulling continues until 500 seconds by which about 95% of interested nodes became seeds. As of the 700 second mark, nodes stop fetching pieces using multi-hop communications. Note that CarTorrent uses the closest-rarest first strategy for piece selection; that is, a piece located in the closest peer is selected for downloading and ties are broken on rarity.

[1]Number of newly born seeds for a given slot.

B. *Impact of mobility*

Next, we investigate the impact of mobility on the download delay. The average download delay as a function of node speed with various node densities is illustrated in Figure 4. We only present the results for the popularity 40% case since the results for other popularity indices show similar trends. (We investigate the impact of popularity in the next subsection.) The figure shows that in CarTorrent as the number of nodes increases, the performance gradually degrades. Given a popularity index, an increased total number of nodes (N) means an increased number of interested nodes, e.g., N = 100 and 200 cases have 40 and 80 interested nodes respectively. As the number of interested nodes increases, the overhead of the underlying routing protocol and gossiping becomes problematic. Fast mobility will induce more route errors especially when the number of interested nodes is large. For instance, when N = 200, the average number of AODV route error messages (RERRs) increases from 61 to 149 when the maximum speed increases from 10 to 30 m/s. Such routes errors will re-initiate route discovery (i.e., RREQ) and will consequently worsen the network congestion. Another important factor attributing to such congestion is the periodical

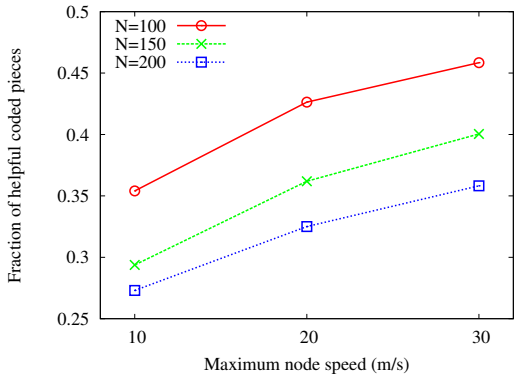

Fig. 6. Helpfulness as function of node speed

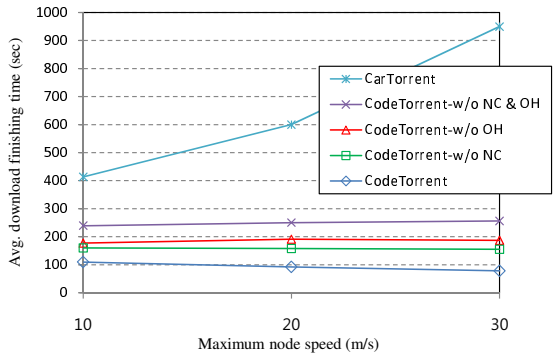

Fig. 7. Comparison of CodeTorrent and its variants (N=200, 40% popularity)

gossiping. Our simulations constrain that a gossiping packet can travel up to 3 hops, and the gossiping period is fixed to 5 seconds. The network congestion is inevitable as the number of nodes participating gossiping increases. Moreover, gossiping period must be adjusted according to mobility in order to accurately choose the closest node (i.e., closest-rarest first selection). That is, the higher the mobility, the more the frequent the advertisements. However, this will further exacerbate the network congestion, thus resulting performance degradation. Figure 5 plots the average number of broadcast packets received per node as a function of node speed and network size in CarTorrent. Broadcast packets composed of routing control packets and gossip messages contribute to overhead. (Recall that data packets are delivered via unicast.) As depicted in the figure, as the network size and mobility increase, the number of broadcast packets sharply increases.

In contrast, the average download delay of CodeTorrent decreases as mobility increases. Since CodeTorrent is based on single hop data pulling and overhearing, mobility plays an important role such that data dissemination latency could be reduced with increased mobility. For ease of an explanation, let us imagine two nodes traveling along the same path without any other contacts until they reach the end of the path. After exchanging useful information at the beginning, the remaining contact period will be useless to each other. We realize that the useless period can only be shortened when we increase their mobility. As shown in Figure 4, this "mobility"-based mixing on top of algebraic mixing through network coding can further reduces the delay. To support this observation, we plot the average fraction of helpful coded pieces (i.e., having a linearly independent code vector) that are pulled or overheard from one's neighbors in Figure 6. As the average number of neighbors increases, it is more probable that a node overhears unhelpful coded pieces from its neighbors. For example, in a static scenario a set of nodes located in between two groups as forwarders, will receive more linearly dependent coded pieces when the size of the target group increases. If nodes are mobile, this can be alleviated, and thus, the helpfulness improves with increased mobility. One caveat is that if the mobility is too high, the contact period will be too short to exchange a piece and this will adversely affect the performance.

The performance advantage of CodeTorrent compared to CarTorrent comes from several aspects. First, CodeTorrent does not incur any routing overhead and the saved bandwidth admits additional data transfer. In CarTorrent, routing and gossip overhead consumes large portion of bandwidth limiting its performance especially when the network size is large and node mobility is high as depicted in Figure 5. Second, the use of network coding helps exploit node mobility and the broadcast nature of wireless medium in full. To identify the source of benefit, we compare CodeTorrent and CarTorrent to its three variants: CodeTorrent without network coding (CodeTorrent-w/o NC), CodeTorrent without overhearing (CodeTorrrent-w/o OH), and CodeTorrent with both network coding and overhearing disabled (CodTorrent-w/o NC & OH) in Figure 7.[2] By comparing CodeTorrent with both network coding and overhearing disabled and CarTorrent, we can see that a major part of delay benefit comes from suppressing routing and gossiping overhead. CodeTorrent with both network coding and overhearing disabled is basically the same as CarTorrent with single-hop communication restriction. Enabling network coding and/or overhearing on top of CarTorrent with single-hop communication restriction further reduces download delay. Enabling both network coding and overhearing at the same time gives 55% to 70% reduction in average download delay whereas enabling just network coding when overhearing is not presented allows only 25% reduction in average download delay. Having said that, we make the following observations from Figure 5 First is that the exploitation of mobility is maximized (compared to other cases) when both network coding and overhearing is enabled. In CodeTorrent, the average download delay is decreased by 30% when the node speed is increased from 10m/s to 30m/s whereas the average download delay is either increased or decreased marginally in CarTorrent and CodeTorrent variants. Second is that the benefit of overhearing is maximized with network coding. When the maximum node speed is 30m/s, the performance improvement from enabling overhearing is 57% when network coding is used (i.e, the performance improvement of "CodeTorrent" from "CodeTorrent-w/o OH"

[2]As in CarTorrent, the closest-rarest first strategy is used for piece selection in the CodeTorrent-w/o NC. All peers are with the same one-hop distance, and thus, a node chooses the least available piece measured in terms of the number of nodes having the piece.

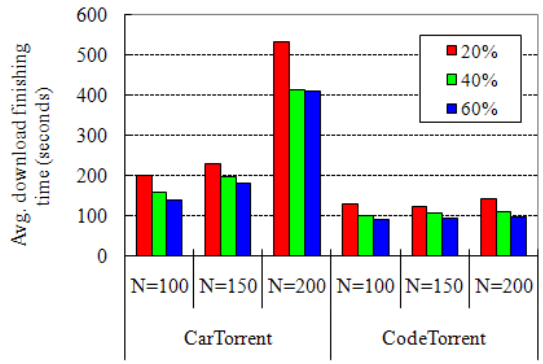

Fig. 8. Impact of popularity on average download delay

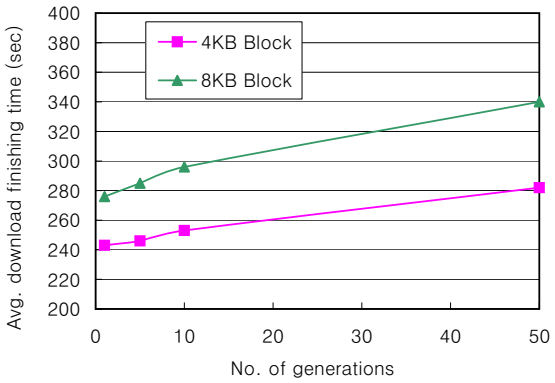

Fig. 9. Download delay as function of number of generations with two different block sizes (4KB/8KB).

is 57%) whereas enabling overhearing provides only 40% performance improvement when network coding is not used (i.e., the performance improvement of "CodeTorrent-w/o NC" from "CodeTorrent-w/o NC & OH" is 40%). Interestingly, when network coding is enabled, the performance gain of overhearing increases, as the node speed increases. This is due to the fact that the usefulness of overheard or pulled coded pieces increases with node speed as shown in Figure 6.

CodeTorrent relies on single-hop communication and thus node mobility is essential to the protocol. CodeTorrent may fail to operate in a static network with a limited number of users with common interest. Dynamically changing topology and/or enough density of users with common interest offers chances for users to communication with each other, i.e., maintains a logical peer-to-peer network, only with single-hop communications. CodeTorrent is designed for MANET in which nodes are moving. We expect that CodeTorrent will show limited performance in the extreme cases with very low node mobility and very few users with common interest.

C. Impact of popularity

To show the impact of the popularity of a file, i.e., the fraction of users who are interested in a specific file, we vary the popularity from 20% to 60% of the population. Figure 8 shows the results with the maximum node speed of 10 m/s. In general, as popularity increases, the average download delay also decreases. In particular, the relative decrement of download delay of CodeTorrent is larger than that of CarTorrent. For example, in CodeTorrent with N=100, we observe 36% and 23% relative decrements when we increase 0.2 to 0.4 and 0.4 to 0.6 respectively, but in CarTorrent with N=100, we see 22% and 12% respectively.

The figure also shows that as node density increases, the average download delay decreases. For a given popularity, we obviously have more interested nodes, as node density increases. More interested nodes in turn mean the increment of availability. As a result, the average download delay can be shortened. In the figure, there is one exceptional case in CarTorrent with N=200 and 60% popularity: unlike all the other cases, as popularity increases, the delay also increases. Again, 60% popularity with N=200 means that there are 120 interested nodes. This implies that there are already too many nodes who want to download the file. These nodes will compete with each other to secure the downloading bandwidth. As a result, there will be severe congestion, and thus, packet drops will increases accordingly. The overall packet loss at the MAC layer can be used to confirm this. The average drops for 20% were 31.4 whereas that for 60% were 168.8 (6 times larger!).

D. Impact of generation size

One of the important performance parameters in network coding based content distribution is the number of generations in the shared file. The use of network coding mitigates the peer and piece selection problem in P2P file swarming. However, once the concept of generation is introduced, the piece (or generation) selection problem reoccurs, i.e., one may find rare pieces or generations, and thus as the number of generations increases the benefit of network coding gets vanished. Another very important benefit of network coding exclusively to the wireless environment comes from the broadcast nature of the wireless medium. When two nodes are exchanging the file pieces the other nodes that can hear their transmission can download the pieces for free. Nodes can exploit overhearing in wireless networks and network coding maximizes the benefit of overhearing. We illustrate the impact of the number of generations using two extreme cases with a file comprised of m pieces: one with a single generation and the other with m generations. In the single generation scenario, any overheard piece is very likely to be useful. Of course, it has to be linearly independent of other downloaded pieces to be helpful. On the other hand, in the m generation scenario the probability that an overheard packet is useful depends on the number of generations that a node has collected so far. When a node has collected k generations, the probability is given as 1 - k/m. The probability decreases as a node collects more generations, that is, the coupon collection problem will happen.

Figure 9 illustrates download delays with different numbers of generations. The results are for the case with 80 interested node out of 200 nodes moving in 30m/s speed and with the 5MB sharing file. For generation selection, a node uses the rarest generation first policy: a node chooses the least available generation measured in terms of the number of nodes having

the generation (i.e., at least one piece.) We use two different piece sizes, either 4KB or 8KB. The large piece size (8KB) cases show larger download delays since, in case of high mobility, file piece transfers fail more frequently. In other words, a large piece takes more time to be transferred but two fast moving nodes crossing each other may not be able to secure contact duration long enough for a successful piece transfer.

Figure 9 clearly shows that as the number of generations increases, the download delay also increases. The single generation case shows 10% to 20% less download delay compared to the case with 50 generations. From the results we can learn that one should decrease the number of generations to achieve better performance. However, there are other issues to be considered. One is the encoding vector size. As we decrease the number of generations, the generation size (the number of pieces belongs to the same generation) and the encoding vector size gets larger and beyond certain point the overhead may become unmanageable. For example, if we have a 10M file divided into 4KB blocks and there is only one generation in the file, the number of blocks in a generation is 2500 and with GF(2^8) the encoding vector size is already over half the block size. Given the file size, to have shorter encoding vectors, one may increase the block size. In our simulations setting however, increasing the block size has a negative effect on the download delay as mentioned above.

The computational overhead of network coding is another issue to be considered. If the decoding process is the bottleneck, the total download delay must include the delay caused by the decoding process. A naive implementation of the decoding process may cost as high as O(m^3) where m is the generation size. However, by employing the progressive/incremental decoding technique proposed in [5], the complexity of decoding can be reduced to $O(m^2)$ [24]. In fact, it is shown in [22] that by using hardware acceleration techniques suited to modern general purpose processors we can decode data at a rate of 16.38Mbps (when m=128) which way exceeds the wireless transmission rate of 2Mpbs. Note that only small portion of the wireless channel bandwidth of 2Mbps is available to each user since the wireless channel is to be shared by nodes in the same collision domain and protocol overhead also consumes some of the bandwidth. Thus, we do not consider the computational overhead in our simulations. If the channel bandwidth is very high and the size of the shared file is also very large, the decoding delay may not be negligible.

IV. Related Work

A major portion of MANET peer-to-peer works can be categorized as mobility-assisted content distribution techniques. A mobility-assisted protocol basically utilizes node mobility to disseminate/retrieve content or index. 7DS [17] aims at sharing web content among nodes based on a high locality of information access within a geographic area, even without Internet connectivity. A node can pull and carry content of interest from its neighbors, thus diffusing content into the network. In Passive Distributed Index (PDI) [13], mobility is exploited for disseminating and maintaining a distributed index of shared content. CodeTorrent is different from these work in that it provides a BitTorrent style content sharing mechanism based on network coding similar to [8]. We show in this paper that there are benefits of using network coding typical to the MANET content sharing scenario, which differentiates our work from [8].

A group of MANET peer-to-peer works is classified as cross-layer techniques. Cross-layer techniques incorporate content indexing and routing as a single layer. Most protocols have been focused on overcoming the discrepancy between a logical overlay and a physical topology of mobile nodes. For example, XL-Gnutella [6] maps the logical overlay neighbors to physical neighbors. CarTorrent [16], a BitTorrent style content sharing protocol in wireless networks, uses the proximity-driven piece selection which is known to perform better than the rarest first piece selection. Similarly, ORION [11] builds an on-demand content-based overlay, closely matching the topology of an underlying network. Unlike these approaches, CodeTorrent attempts to tackle dynamically changing topology and intermittent connectivity in MANET as well as various other issues that have been disregarded in previous mobile peer-to-peer researches such as addressing, node/user density, non-cooperativeness, and unreliable channel all together with the help of network coding and mobility assisted dissemination techniques.

Also cross-layer approaches can be found in in distributed hash table (DHT) inspired protocols. The VRR protocol is a clean slat approach for Content Based Routing in MANET. VRR is implemented on the top of the link layer and provides a reliable routing as well as a native content addressable network [3]. Another DHT inspired approach is represented by the Geographic Hash Tables [21] where the contents hare hashed into a geographic location and the retrieval is driven by the a geographic routing performed over the MANET.

Using network coding in P2P was first proposed in [8] and recent feasibility studies on network coding in real testbeds have been done in [7], [14], [23]. Various performance enhancement techniques have been proposed [14], [15], [22]. In [14], authors propose the sparse network coding where each piece is selected for coding with a certain probability, thus reducing the number of pieces involved in coding. Maymounkov et al. show that one can decrease the generation size, yet can still effectively handle the coupon collection problem by using erasure codes at the generation level [14]. The coding technique proposed in [14] may alleviate the generation selection problem in our protocol. The technique allows the nodes to collect any m generations out of total k (bigger than m) generations instead of collecting each of m distinct generations. However, even with the technique in place, different node pairs still need to exchange coded frames belonging to different generations reducing their chances to be helpful to each other. Thus, we expect that employment of the coding technique proposed in [14] instead of RLC will not lead to meaningful reduction in download delay with our protocol especially under the simulation settings used in this paper. Shojania et al. in [22] use CPU acceleration techniques to improve the performance of Galois Field operations. Some

other network coding researches relevant to our scenario are the opportunistic routing with network coding [10] and multicasting [18] in the wireless environment and network coding based message dissemination in delay tolerant networks [25].

V. Conclusion

In this paper we proposed a network coding based file swarming protocol named CodeTorrent offering less download delay than an existing file swarming protocol. We showed that network coding helped exploit unique opportunities offered in the MANET P2P environment, the broadcast nature of wireless medium and node mobility and by exploiting them in full we could have a very simple solution to the issues arising in MANET file swarming systems. We kept the system in the simplest form in this paper for clearer presentation of the main idea. Immediate future work includes exploring optimization opportunities that the proposed protocol allows.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. Inform. Theory*, 46(4):1204–1216, July 2000.

[2] F. Bai and A. Helmy. Impact of Mobility on Mobility-Assisted Information Diffusion (MAID) Protocols. Technical report, USC, July 2005.

[3] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: Network routing inspired by dhts. In *Proc. ACM SIGCOMM 2006*.

[4] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan. Can network coding help in p2p networks? In *Proc. NetCod06*.

[5] P. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proc. Allerton*, 2003.

[6] M. Conti, E. Gregori, and G. Turi. A Cross-Layer Optimization of Gnutella for Mobile Ad hoc Networks. In *Proc. ACM MobiHoc05*.

[7] C. Gkantsidis, J. Miller, and P. Rodriguez. Comprehensive view of a live network coding p2p system. In *Proc. IMC06*.

[8] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *IEEE INFOCOM05*, 2005.

[9] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Trans. Inform. Theory*, 52(10):4413–4430, Oct. 2006.

[10] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard. The importance of being opportunistic: Practical network coding for wireless environments. In *Proc. Allerton05*.

[11] A. Klemm, C. Lindemann, and O. Waldhors. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In *Proc. VTC 2003-Fall*.

[12] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Trans. Networking*, 11(5):782–795, Oct. 2003.

[13] C. Lindemann and O. Waldhors. A distributed search service for peer-to-peer file sharing in mobile applications. In *Proc. IEEE P2P 2002*.

[14] G. Ma, Y. Xu, M. Lin, and Y. Xuan. A content distribution system based on sparse linear network coding. In *Proc. NetCod07*.

[15] P. Maymounkov, N. J. A. Harvey, and D. S. Lun. Methods for efficient network coding. In *Proc. Allerton06*.

[16] A. Nandan, S. Das, G. Pau, M. Sanadidi, and M. Gerla. Cooperative downloading in vehicular ad hoc wireless networks. In *Proc. WONS'05*.

[17] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices. In *Proc. ACM MOBIHOC 2001*.

[18] J.-S. Park, D. S. Lun, Y. Yi, M. Gerla, and M. Medard. Codecast:a network-coding-based ad hoc multicast protocol. *IEEE Wireless Communications*, 13(5), 2006.

[19] C. E. Perkins and E. M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Proc. IEEE WMCSA'99*.

[20] Scalable Networks. http://www.scalable-networks.com.

[21] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A geographic hash table for data-centric storage in sensornets. In *Proc. WSNA'02*.

[22] H. Shojania and B. Li. Parallelized progressive network coding with hardware acceleration. In *Proc. IWQoS*.

[23] M. Wang and B. Li. How practical is network coding. In *Proc. IWQoS*.

[24] M. Wang and B. Li. Lava: A reality check of network coding in peer-to-peer live streaming. In *Proc. IEEE INFOCOM07*.

[25] J. Widmer and J.-Y. L. Boudec. Network coding for efficient communication in extreme networks. In *Proc. CHANTS05*.