Efficient Multipoint P2P File Sharing in MANETs

Afzal Mawji and Hossam Hassanein
Telecommunications Research Lab, School of Computing, Queen's University
Kingston, Ontario, Canada, K7L 3N6
{mawji, hossam}@cs.queensu.ca

Abstract—**Peer–to–peer networks are a popular means of obtaining large files. Network coding has been shown as an efficient means of sharing large files in a P2P network. With network coding, all file blocks have the same relative importance. This paper presents Deluge, which uses network coding to share large files in a P2P overlay running on a MANET. Peers request file blocks from multiple server nodes and servers multicast blocks to multiple receivers, providing efficient multipoint–to–multipoint communication. Simulation results show that compared to other common download techniques, Deluge performs very well, having low download time and low energy consumption. Further, more peers participate in uploading the file and the performance of Deluge varies little with increasing overlay size, indicating good scalability.**

I. Introduction

Peer–to–peer (P2P) networks are immensely popular among users and their uses vary over many different application types, such as file sharing, Voice–over–IP, gaming and instant messaging. Combining mobile ad hoc networks (MANETs) and peer–to–peer networks together so that a P2P overlay runs on a cooperative MANET seems natural, since they share many similarities: both are fully decentralized, must dynamically organize, must deal with frequent topology changes, must be resilient to failure, and must perform the routing function.

It is clear that users want to run P2P applications, regardless of what type of network they are using, be it a wired network, a MANET, or a hybrid infrastructure–infrastructureless network such as a wireless mesh network. Therefore, it is important to consider how to effectively run a P2P overlay in an infrastructureless network such as a cooperative MANET. Even if the user is connected to a partly–infrastructured network such as a mesh network, avoiding the base station and instead communicating with nearby nodes, where possible, may reduce delay and energy use since transmissions would travel a shorter distance and thus require less power. Furthermore, avoiding communication with the base station may avoid charges from the network operator.

Though P2P networks and MANETs share many similarities, it seems that simply adopting existing P2P overlay techniques and using them in MANETs is undesirable, since there also signicant differences. P2P networks are very large–scale with millions of simultaneous users, and are designed as overlays for deployment on the "edge" of the Internet, where the nodes generally do not move about. MANETs, on the other hand, tend to have far fewer nodes that are geographically nearby one another, the devices are severely resource–constrained in comparison, particularly in terms of energy, and the links between nodes usually have higher delay.

Network coding is a technique designed to save bandwidth, and which therefore allows more efficient downloads in a P2P networks. Network coding removes the "rarest block" problem, in which a particular file block is difficult to find and requires lengthy waiting once located. With network coding all blocks are of equal value. This allows nodes to download without searching for specific blocks, increasing robustness and reducing download time.

This paper examines the effects of using network coding in a P2P file sharing overlay in a cooperative mobile ad hoc network. To our knowledge, it is the first such paper to do so. Because there is no centralized authority and no infrastructure, a tracker node cannot be used, as is commonly done in P2P networks. Instead we make use of multicasting for efficient communication within the overlay. The algorithm presented, called Deluge, uses the idea of a server node multicasting blocks, something that is impractical on the Internet, but is possible in MANETs. Clients request a certain number of blocks from multiple different servers, depending on the cost of acquiring them, resulting in multipoint–to–multipoint communication. In this paper, cost is measured as hop distance. Therefore, nodes download files from the closest servers, which reduces overall network energy consumption.

In this paper, the term node refers to a device in the MANET that may or may not be a part of the P2P overlay. The terms peer and overlay member are used interchangeably and refer to a device in the MANET that is part of the P2P overlay. A peer is a neighbor if there is a direct connection to it in the overlay. The word server is used for nodes that have at least one encoded block and are able to upload to other nodes. The term client is used for peers that are downloading or wish to download a file in the overlay.

Simulation results show that compared to downloading the entire file from a single server, or downloading uncoded blocks from multiple servers, Deluge reduces download time and also significantly reduces energy consumption. In addition, more nodes upload blocks, resulting in a fairer distribution of the load. Finally, Deluge's performance tends to remain relatively constant regardless of overlay size, indicating stable performance as the number of overlay members scales upward.

The rest of the paper is organized as follows. Section II briefly describes network coding and examines its use in P2P and wireless networks. Section III describes Deluge in detail. Section IV presents our simulation results and analysis and Section V gives our conclusions.

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

II. Background

Network coding was introduced by Ahlswede *et al.* [1] and linear network coding was introduced by Li *et al.* [2] as a technique to save bandwidth. There are several tutorial papers covering practical network coding [3]–[5]. Network coding is a form of information spreading, in which nodes, instead of simply forwarding data packets, combine, or encode, several packets together using the XOR operation. The idea is to shift some of the work burden from the network to the nodes' computational abilities. This is a good tradeoff since network bandwidth is generally more expensive, i.e. slower, than computation. One major benefit of network coding is that encoded packets can be further encoded. This allows nodes to encode data without first decoding it, so that they can encode packets for which they do not yet have the completed data.

The linear combination of data, interpreted as a set of numbers over a finite field, is then transmitted in place of the original packets, reducing the amount of data that must be sent, and thus increasing throughput. The data of size s is considered as a symbol over the field $\mathbb{F}_{2^s}$. The original packets are associated with a set of coefficients over the field. These coefficients are multiplied by the original packet, s bits at a time, and XORed with the same bit positions of the other packets to be combined. The encoded data, known as the information vector, is sent along with the set of coefficients, known as the encoding vector. The encoding vector is needed by the receiver to decode the data. The coefficients are selected at random, in a completely independent and decentralized manner. It is believed that even with a small field size of $s = 8$ the probability of selecting linearly dependent combinations is negligible [4].

Nodes receiving the encoded packets do not need to worry about obtaining specific packets. Instead, they must obtain a sufficient number of linearly independent, or "innovative", packets. The encoded packets, along with the given encoding vectors, are considered as a system of equations, where the original messages are the unknowns. For a set of n original messages and m received messages, we therefore have m equations with n unknowns, and so we must have $m \geq n$ to solve the system. Furthermore, there must be at least n linearly independent equations. Any known technique, such as Gaussian elimination, may be used to solve the system and thus recover the original data.

Katti *et al.* present the first implementation of practical network coding to the wireless environment [6], [7]. The approach, called COPE, is for nodes to listen and remember what packets neighbors have received so that the "best" source packets are encoded to minimize the number of transmissions. This opportunistic approach relies only on local information and exploits coding opportunities in real-time. All nodes in the wireless network participate. Transmitted packets are annotated to inform neighbors which packets the transmitter has heard. When a node sends data, it uses the knowledge of what its neighbors have received to perform opportunistic coding. Simulation results as well as multi-node test beds confirm the benefits of network coding.

Gkantsidis *et al.* bring practical network coding to the realm of P2P file sharing [8]–[10]. Their system, dubbed Avalanche, is designed to allow nodes to more quickly download a large file. The source node along with other peers that have parts of the file, encode all the blocks they have available. As long as a downloading peer gets enough linearly independent blocks, it can decode them to retrieve the entire file. The authors show that even if the server leaves shortly after seeding one copy, the network will still be able to get a high completion rate for the file. They also show that the CPU and I/O requirements of encoding and decoding add minimal overhead. Avalanche assumes the use of a centralized tracker which coordinates the nodes. In a MANET, such a node is impractical. Furthermore, Avalanche is designed for wired networks and does not examine energy use.

III. Deluge

Avalanche, the main previous work on using network coding for P2P file sharing [8]–[10], encodes a file across all blocks that a peer currently has. Similarly, Deluge also encodes files across all blocks that a peer currently has. However, Avalanche assumes the use of a tracker node to coordinate peers. In a MANET, there are no centralized nodes and so trackers are impractical. Instead Deluge relies on a greedy algorithm to choose which servers to download blocks from.

A. Multipoint Download

In this paper, we focus only on the file download function and assume that when a client wishes to download a file, it has already run a search query and possesses results, which include a list of servers that have at least one encoded block constituting the file and also the value of the cost function for each server. For the purposes of this paper, we assume the cost function is the hop distance, which is easily available from the routing protocol. A closer node is preferred to a further one because it would reduce the cost to the network. In the event that some incentive system using credits is in place, for example, clients could choose servers with the lowest credit price.

Given the list of servers, block counts, and hop distances, the client uses a greedy algorithm to determine from whom to download, and how many blocks to request from each server. Because the blocks are encoded, all blocks have the same level of importance and the peer must simply download enough innovative blocks. Note that intermediate routing nodes merely forward data between source and destination; they are not involved in the encoding process, nor do they store encoded packets.

The steps taken by a client to download a file are as follows. A counter, b, is initialized with the total number of blocks the file to be downloaded has been split into. Next, the list of servers is sorted based on cost, with the lowest cost server appearing first. The server at the front of the list, s, is then sent a download request. The number of blocks requested is the minimum of the number of blocks remaining to be downloaded, b, and the number of blocks the server has, n_s. This number is then deducted from the counter, and if more blocks are needed, the next server on the list is contacted.

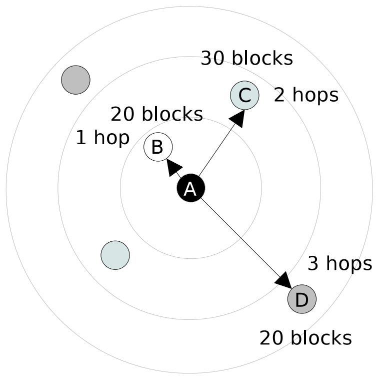

Fig. 1. A client downloading blocks from the nearest servers

Ties are broken arbitrarily. The algorithm results in the client requesting all of the blocks of the file from the servers such that the cost is minimized.

As mentioned earlier, in this paper the cost function is hop distance. Therefore, the closest server is contacted first and as many blocks as are available are downloaded from it. If more blocks are required, the next closest server is contacted, and so on. This is illustrated in Fig. 1. In the figure, Node A is attempting to download a file which consists of 60 blocks. Node B, within 1 hop, is contacted first and 20 blocks are requested from it, since those are all the blocks B has. Node C, 2 hops away, is then contacted, and 30 blocks are requested from it. Finally, node D, which is 3 hops distant, is contacted. Only 10 blocks remain, and so 10 blocks are requested from D instead of the 20 it has available. The unlabeled nodes do not have any blocks of the file.

It may happen that due to client, server, or intermediate node mobility, or the receipt of too many non–innovative blocks, a client may not be able to download the entire file as expected. In this event, the client repeats the search query to get an updated list of servers and costs, and then re-runs the algorithm, but leaves the counter, b, as is. This allows the client to obtain the remaining number of blocks required.

B. Multipoint Upload

When a server receives a request for blocks, it sets up a multicast group and informs the requesting peer of its address. If the server was already uploading blocks, it sends the address of the existing group. All blocks are sent via multicast. In this way, a single send can result in the block being received by multiple clients. Clients download blocks from multiple senders simultaneously, and servers upload blocks to multiple receivers. The resulting multipoint–to–multipoint communication is very efficient, reducing energy costs and also download time. Any multicast routing protocol may be used.

The steps a file serving peer performs when it receives a download request are as follows. The incoming request from the client contains the requested number of blocks, n. The server keeps a counter of blocks that have been requested to be sent, b. Since all the blocks it sends are encoded, the blocks themselves are indistinguishable. When a request comes in, the counter is incremented by the number of blocks requested. The multicast group address is sent to the client and the server begins or resumes multicasting blocks.

Once a server has finished submitting all blocks, i.e. the counter b reaches zero, it ceases sending and makes the multicast group address available for use by others. Download clients, when they have completed downloading the file, simply leave the multicast group. There is no need to inform the server that they have finished downloading, since the server keeps its own counter of blocks remaining to be sent.

IV. Simulation Results and Analysis

This section evaluates the performance of Deluge for different P2P overlay sizes and compares it to the performance of downloading the entire file from a single server (referred to Single Server) and using unencoded blocks (referred to as Unencoded). Note that Single Server means that peers attempt to download the entire file from one server at a time. For Unencoded and Deluge, clients may download from multiple servers simultaneously. Single Server and Unencoded are common means of downloading files is modern P2P networks.

A. Simulation Parameters and Metrics

The MANETs are simulated in *ns–2* 2.33, have an area of 1500m $\times$ 1500m, and contain 100 nodes. A transmission rate of 54 Mbps is used. All nodes are evenly distributed in the simulation area.

The random waypoint model is used for mobility with nodal velocities distributed according to a uniform distribution with minimum speed of 1 m/s and maximum speed of 3 m/s. The pause time is uniformly distributed with a mean of 60 seconds and the simulation time is set to two hours. MAODV is used as the multicast routing protocol and AODV is used as the unicast routing protocol.

In each simulation experiment the number of nodes participating in the P2P overlay is varied, with the range being 30 nodes to 100 nodes. Experiments using Deluge proceed as described in Section III. Experiments for downloading using Single Server prescribe that a client locate the nearest server that has the entire file and initiate a download from it. In the event of an interruption, downloads are resumed instead of starting over. Experiments for downloading using unencoded blocks download the blocks in "rarest first" order, from the nearest peer that has the block. Rarest first means that a node will attempt to download the block that has the fewest copies distributed in the network.

Each experiment begins with a single, randomly selected overlay member, the initial server, having the complete file. All other nodes in the overlay have no piece of the file at all. The file is reasonably large, at 100 MB, and is broken into 1000 blocks. Every 60s of simulation time, a new, random peer will attempt to download the file using the method for that experiment (i.e. Deluge, Single Server, or Unencoded). If a client is unable to complete a download, due to node mobility, it will try downloading the file again. Several runs were completed for each experiment with the results averaged.

TABLE I
ENERGY CONSUMPTION CONSTANTS USED IN SIMULATION

m_{send}	1.89	$mW \cdot sec/byte$
b_{send}	246	$mW \cdot sec$
m_{recv}	0.494	$mW \cdot sec/byte$
b_{recv}	56.1	$mW \cdot sec$
$b_{sendctl}$	120	$mW \cdot sec$
$b_{recvctl}$	29.0	$mW \cdot sec$

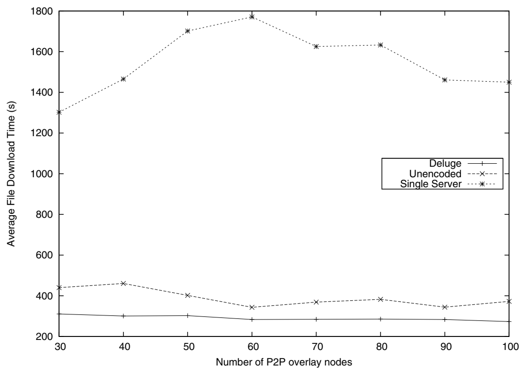

Fig. 2. Average file download time

TABLE II
DOWNLOAD TIME IN SECONDS FOR 100 OVERLAY NODES

	Average	Max	Min	Std. Dev.
Deluge	273	2,113	17	290
Unencoded	373	9,330	17	703
Single Server	1,450	6,640	17	1,364

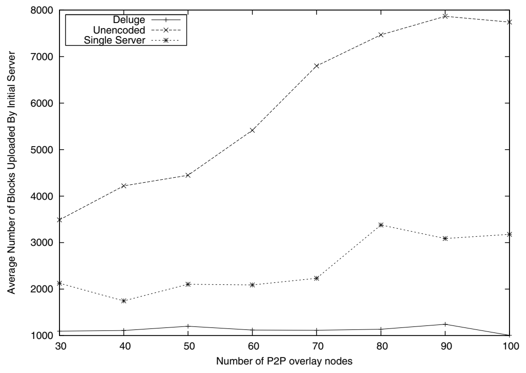

Fig. 3. Average number of blocks uploaded by initial server

The metrics considered are download time, energy consumption of peers and the initial server, and average number of blocks uploaded by peers and the initial server. If a node does not complete downloading the file, its information is not included in the averages, since this would skew the energy consumption and download times. Only the Single Server experiment type had failed downloads.

The energy consumption model used in the simulations is the linear model proposed by Feeney [11]. Each MAC layer operation takes a certain amount of power as defined by $cost = m \times size + b$ where m is the incremental cost of the operation, b is the fixed cost, and $size$ is the amount of data sent or received. The constants are obtained by physical measurements for a Lucent IEEE 802.11 WaveLAN PC Card from [11] and are summarized in Table I. The multicast overhead is included in the measurements.

B. Simulation Analysis

Of significant importance to users in a P2P network is the amount of time it takes to download a desired file. Fig. 2 shows the average time it takes to download the file in seconds. Deluge has the lowest download time and it does not vary much as the overlay size increases. Unencoded also has a relatively constant download time as overlay size increases, however it is higher than Deluge. This is because the the rarest block is more difficult to obtain than the others and increases the download time. Single Server has the slowest download time by far because clients are only able to download from one server at a time. Also, until the entire file has been uploaded at least once, peers have no choice but to attempt the download from the initial server. The time to download increases with the network size until it reaches 60 nodes, after which it decreases because there are more nodes in the network and so more servers are available for the later nodes to choose from.

The average download times shown in Fig. 2 conceal the immense variation in download time that peers experience for the different download systems tested. Table II provides additional information for the case of 100 overlay nodes. It can be seen that though the Deluge and Unencoded average download times are not significantly different, the maximum download time and standard deviation are. Single Server also varies immensely for the reasons discussed earlier.

When sharing a file in P2P overlays, it is desirable, from the network's point of view, for as many nodes as possible to contribute in uploading the file. Sharing the burden of uploading is fairer, and reduces the load on the initial server. From an energy viewpoint, it also spreads energy use through the overlay so that no node uploads so much that it runs out of energy too quickly.

Fig. 3 shows the average number of blocks uploaded by the initial server. The server must upload at least 1000 blocks so that one complete file is available in the network. Deluge servers uploaded only slightly more than the minimum number of blocks regardless of overlay size, demonstrating the usefulness of network coding. Unencoded servers uploaded the equivalent of several copies of the file on average, and as the number of peers increased, the burden on the initial server increased significantly as well. This is because the initial server always possesses a copy of the rarest blocks. Even when all blocks have been uploaded at least once, the rarest ones are still available, though not exclusively, from the initial server. Single Server's initial server also uploaded multiple copies of the file with the number of blocks uploaded increasing with overlay size. This is because all peers must download the entire file from it until at least one complete copy of the file has been downloaded.

Fig. 4 shows the average number of blocks uploaded per node, not including the initial server node. Deluge peers generally uploaded about 80% of a complete file, but importantly this number does not change appreciably as the number of peers increases. Unencoded uploads more blocks, and rises slightly with more nodes. This is despite the fact that the Unencoded initial server uploads far more blocks as compared

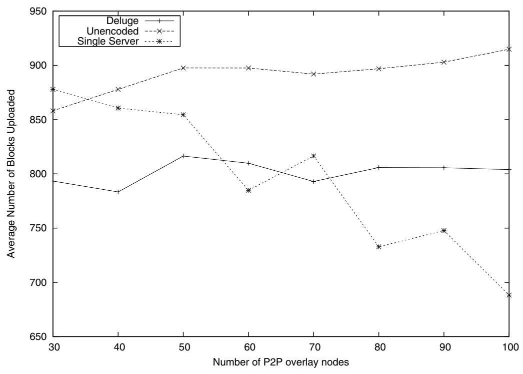

Fig. 4. Average number of blocks uploaded

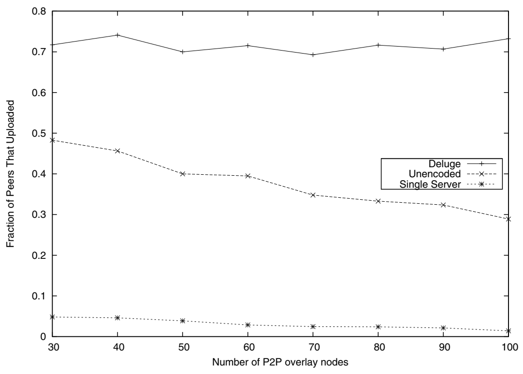

Fig. 5. Fraction of nodes which uploaded at least one block

TABLE III
BLOCKS UPLOADED BY INITIAL SERVER FOR 100 OVERLAY NODES

	Average	Max	Min	Std. Dev.
Deluge	1,006	1,032	1,000	11
Unencoded	7,742	17,845	2,222	4,398
Single Server	3,178	11,590	1,000	3,300

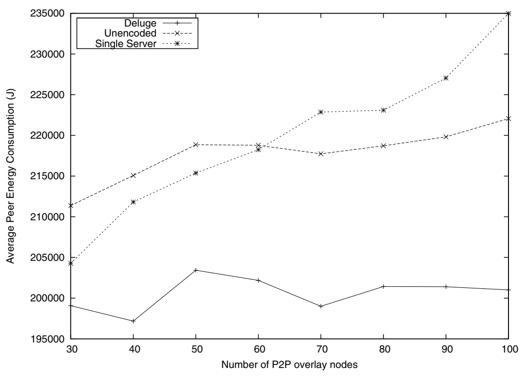

Fig. 6. Average energy consumption in Joules

to Deluge. Single Server peers' uploaded blocks decreases as the number of overlay members increases, since a small number of servers handle most of the load.

Tables III and IV provide additional information about the number of blocks uploaded for the case of 100 overlay nodes. Here it can be seen that Deluge, in addition to having better average numbers, also varies less across all peers. For the initial server case, Deluge uploaded barely more than a single completed file, whereas the other two systems uploaded multiple copies, and had a standard deviation of 3–4 full files across the experimental runs.

Fig. 5 shows what fraction of peers uploaded at least one block of data. In the case of Single Server, generally a server will upload the entire file to a client, but due to node mobility this may not occur, and another node would be required to resume the file upload. Unsurprisingly, Single Server had the fewest peers participate in uploading a file because until later in the experiment, relatively few nodes would have the entire file available to upload. Less than half of the Unencoded peers uploaded blocks, and this number decreased as the overlay increased in size. This happens because new downloading peers search for the rarest block first, and until the entire file has been uploaded at least once, the rarest block is always one that has not yet been uploaded, and exists only with the initial server. Therefore, the initial server must do all of the uploading to all clients until it has completely uploaded the entire file. In contrast, over 70% of Deluge nodes contributed to uploading the file and this number was relatively constant regardless of overlay size. Because network coding removes the relative importance of blocks, any block from any node will do, as long as it is linearly independent of all received blocks.

TABLE IV
BLOCKS UPLOADED PER PEER FOR 100 OVERLAY NODES

	Average	Max	Min	Std. Dev.
Deluge	804	8,904	0	1,411
Unencoded	915	18,607	0	2,475
Single Server	688	79,296	0	6,764

Fig. 6 indicates the energy use of peers, not including the initial server. Once again Deluge peers show relatively constant use, regardless of overlay size. This is the case because all file blocks are roughly equivalent, and so any server is capable of uploading a block, which distributes the load among all peers. Unencoded and Single Server show increasing energy use with overlay size because as more peers exist within the overlay there is more demand on the small number of servers which have blocks to upload. These few nodes have a heavy upload burden, while the majority of nodes do not upload anything at all.

The average energy consumption data also conceals the immense variation in energy usage. Table V gives additional information, showing energy consumption data for the case of 100 overlay nodes. We can see that all three systems had the same minimum energy usage, from a node adjacent to a server and which did not upload any blocks; but the maximum and standard deviation vary significantly. In the experiments carried out, nodes were not assigned a maximum energy capacity. If they had been, any reasonable capacity would have resulted in many nodes running out of energy, which would have had further negative repercussions on other nodes' energy use, as well as download time. In light of this, it is likely that Deluge would have even further outperformed its competing systems.

Table VI gives additional information about the energy consumption of the initial server nodes for the 100 overlay member case. Again, Deluge has performed exceptionally

TABLE V
ENERGY USED IN JOULES PER PEER FOR 100 OVERLAY NODES

	Average	Max	Min	Std. Dev.
Deluge	201,031	1,731,790	49,120	266,765
Unencoded	222,059	3,464,307	49,120	467,884
Single Server	234,986	15,038,364	49,120	1,525,233

TABLE VI
ENERGY USED IN JOULES PER INITIAL SERVER FOR 100 OVERLAY NODES

	Average	Max	Min	Std. Dev.
Deluge	190,197	195,048	189,000	2,088
Unencoded	1,463,526	3,373,223	420,023	831,285
Single Server	600,734	2,190,846	189,029	623,796

well, with little variation over all experiments, while the other two systems had significant variation. Energy use is so high for Unencoded in particular, that it is likely the initial server would have run out of energy before even a single completed copy of the file had been uploaded into the network. Deluge's use of multicasting when uploading blocks has significantly reduced its energy consumption.

Fig. 7 shows the number of incomplete downloads. Both Deluge and Unencoded were always able to have all peers complete downloading in the given simulation time. Therefore they are not shown in the figure. However, Single Server had nodes with incomplete downloads, and as the overlay size increased, the number of incomplete downloads increased rapidly. This is due to the overwhelming burden of uploading blocks placed on the initial server. Too many download requests resulted in nodes not being able to receive the whole file, which in turn meant they could not upload for later peers. With 100 overlay nodes, nearly $\frac{1}{3}$ of peers were unable to complete the file download. These peers with incomplete downloads were not included in the data shown earlier.

V. CONCLUSIONS

This paper presented Deluge, a system that uses network coding for file downloads in a P2P overlay running on a mobile ad hoc network. Network coding removes the relative importance of file blocks and allows more efficient use of bandwidth. Servers encode across all blocks they possess and multicast the encoded blocks so that multiple clients can receive blocks from a single transmission. Peers choose to download blocks from multiple servers such that their cost is minimized, resulting in efficient multipoint–to–multipoint communication.

Simulation results demonstrate that Deluge clearly outperforms the commonly used download system of unencoded blocks in which peers download from multiple servers and attempt to get the rarest block first. Deluge also performed significantly better than another common system in which a server uploads the entire file to a peer. Deluge had lower average download time, lower energy usage, spread the upload burden amongst peers better, and did not have any incomplete download attempts. Also, Deluge's performance tended to vary little, even as the overlay grew in size. This indicates that Deluge may scale very well to large overlay networks. In contrast, Unencoded and Single Server peers used significant amounts of energy and experienced wide variation in their performance. Furthermore, matters tended to worsen for them as the overlay grew in size.

In the future, we will add the capability for intermediate nodes to temporarily store coded blocks and upload them in response to download requests they forward. This would likely result in even lower download times for most users, though the number of non–innovative blocks received may increase. We will also integrate Deluge into a complete P2P system, which includes the file query and response mechanism. Finally, we will look at extending our work to wireless mesh networks, which incorporate some infrastructure elements.

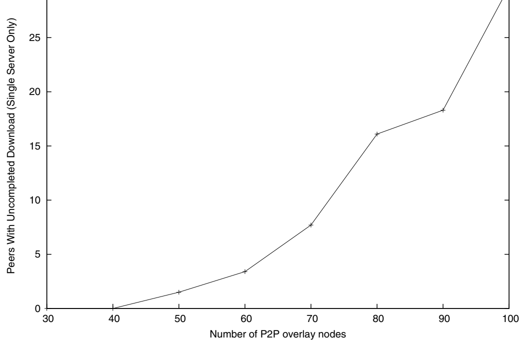

Fig. 7. Average number of incomplete downloads

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.
[3] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton Conference Communication, Control and Computing*, October 2003.
[4] C. Fragouli, J.-Y. L. Boudec, and J. Widmer, "Network coding: An instant primer," *Computer Communication Review*, vol. 36, no. 1, pp. 63–68, January 2006.
[5] P. A. Chou and Y. Wu, "Network coding for the internet and wireless networks," *IEEE Signal Processing Magazine*, pp. 77–85, September 2007.
[6] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Médard, "The importance of being opportunistic: Practical network coding for wireless environments," in *Allerton Conference Communication, Control and Computing*, 2005.
[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 243–254, October 2006.
[8] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *IEEE Infocom*, March 2005, pp. 2235–2245.
[9] C. Gkantsidis, J. Miller, and P. Rodriguez, "Anatomy of a p2p content distribution system with network coding," in *International workshop on Peer-To-Peer Systems (IPTPS)*, February 2006.
[10] ——, "Comprehensive view of a live network coding p2p system," in *ACM SIGCOMM conference on Internet measurement*, October 2006, pp. 177–188.
[11] L. M. Feeney, "An energy–consumption model for performance analysis of routing protocols for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 6, no. 3, pp. 239–250, 2001.