A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks

Alexander Klemm, Christoph Lindemann, and Oliver P. Waldhorst
Department of Computer Science
University of Dortmund
August-Schmidt-Str. 12
44227 Dortmund, Germany
http://www4.cs.uni-dortmund.de/~Lindemann

Abstract — **Establishing peer-to-peer (P2P) file sharing for mobile ad hoc networks (MANET) requires the construction of a search algorithm for transmitting queries and search results as well as the development of a transfer protocol for downloading files matching a query. In this paper, we present a special-purpose system for searching and file transfer tailored to both the characteristics of MANET and the requirements of peer-to-peer file sharing. Our approach is based on an application layer overlay network. As innovative feature, overlay routes are set up on demand by the search algorithm, closely matching network topology and transparently aggregating redundant transfer paths on a per-file basis. The transfer protocol guarantees low transmission overhead and a high fraction of successful downloads by utilizing overlay routes. In a detailed ns-2 simulation study, we show that both the search algorithm and the transfer protocol outperform off-the-shelf approaches based on a P2P file sharing system for the wireline Internet, TCP and a MANET routing protocol.**

Keywords — Data distribution and replication, overlay networks, content-based routing, transport in wireless networks.

I. INTRODUCTION

Mobile ad hoc networks (MANET) and peer-to-peer (P2P) file sharing systems both exhibit a lack of fixed infrastructure and possess no a-priori knowledge of arriving and departing peers. Due to this common nature, P2P file sharing seems natural and attractive to be deployed for MANET. Interesting application scenarios include sharing traffic and weather data by car-to-car communication in a wide-range MANET, a groupware system for mobile e-learning applications in a local-range MANET running on IEEE 802.11, and sharing music, jingles, video clips etc. from mobile device to mobile device via Bluetooth.

The operation of most P2P systems in the wireline Internet depends on application layer connections among peers, forming an application layer overlay network. In general, these connections are static, i.e., a connection between two peers remains established as long as both peers dwell in the system. We identify the maintenance of static overlay connections as the major performance bottleneck for deploying a P2P file sharing system in a MANET. We find that the overlay network topology does not reflect the underlying MANET topology, neither in terms of connection layout nor in connection lifetime. Whereas the overlay network topology is static in the timescale of a node's dwell time in the system, the MANET topology changes much more frequent due to node mobility. This induces significant control overhead for connection maintenance, resulting in increasing network traffic and decreasing search accuracy.

In this paper, we present a special-purpose approach for P2P file sharing tailored to MANET denoted Optimized Routing Independent Overlay Network (ORION). ORION comprises of an algorithm for construction and maintenance of an application-layer overlay network that enables routing of all types of messages required to operate a P2P file sharing system, i.e., queries, responses, and file transmissions. Overlay connections are set up on demand and maintained only as long as necessary, closely matching the current topology of the underlying network. ORION combines application-layer query processing with the network layer process of route discovery, substantially reducing control overhead and increasing search accuracy compared to an off-the-shelf approach utilizing a P2P system for the wireline Internet, TCP, and a state-of-the-art MANET routing protocol. Additionally, the overlay network enables low overhead file transfer, as well as an increasing probability for successful file transfers compared to the off-the-shelf approach. Performance gains with respect to the off-the-shelf approach are illustrated in a simulation study.

ORION operation does not depend on the deployment or support of any MANET routing protocol. Note that some data structures and mechanisms used by ORION are also provided by reactive MANET routing protocols, e.g., AODV [9] or DSR [5]. I.e., ORION might induce some redundancy or even duplication when deployed on top of such protocol. However, we would like to point out that the basic concepts of ORION could be combined with routing protocol functionality to avoid duplication and make use of existing synergies.

The remainder of this paper is organized as follows. Section II summarizes related work in the area of P2P systems. In Sections III and IV, we present the ORION search algorithm and transfer protocol. In Section V, the performance of ORION is compared to an off-the-shelf approach in a simulation study. Finally, concluding remarks are given.

0-7803-7954-3/03/$17.00 ©2003 IEEE.

II. RELATED WORK

Most published P2P file sharing systems have been introduced for the wireline Internet. In general, a P2P file sharing system consists of two building blocks: (1) a search algorithm for transmitting queries and search results and (2) a file transfer protocol for downloading files matching a query. Whereas most file sharing systems transfer files directly between peers using TCP connections, efficient searching is an active area of research for wireline P2P systems.

The P2P system Gnutella [7] released by AOL in 2000 constitutes the first system implementing a fully distributed file search. Queries are broadcasted to all peers using an application-layer overlay network. Despite its poor performance, Gnutella gained rapidly increasing popularity. Several recent approaches substantially improved the scalability of the query process by reducing the number of messages generated to resolve a user query. Aberer et al. propose P-Grid, a virtual binary search tree, to route query messages to a number of nodes, which are responsible to answer these queries [1]. Each peer in a P-Grid has to maintain application-layer connections to two other peers. Other state-of-the-art P2P systems like CAN [11] and Chord [12] implement distributed hash tables. These systems allow queries for keys, which are resolved by routing the query to a peer storing the value matching this key. For systems with n peers, a query can be resolved involving $O(\log n)$ (or $O(n^{\alpha})$ for $\alpha < 1$) intermediate routing hops. Each node in CAN has to maintain connections to $2d$ neighbors, where d is a configuration parameter. Chord maintains connections to $O(\log n)$ neighbors. Thus, like Gnutella, all state-of-the-art P2P systems build overlay networks with static connections between participating peers. Opposed to the P2P systems mentioned above, the approach presented in this paper does not employ static overlay connections, but sets up connections on-demand, i.e., during query processing. Thus, application layer routes closely match the current network topology, significantly reducing overhead.

A first approach for establishing P2P file sharing for MANET constitutes the protocol 7DS [10]. 7DS uses local broadcast transmissions for sharing Web documents among peers in order to enable online Web-browsing without connection to the Internet. In a recent paper [8], the concept of Passive Distributed Indexing (PDI) was introduced. Both approaches concentrate on the search algorithm of a file sharing system for MANET. The innovation of 7DS and PDI lies in replacing traffic-intensive routing by exploiting peer mobility. Opposed to 7DS and PDI, we investigate in this paper the exploitation of forwarding and routing capabilities of a MANET for P2P file sharing systems.

III. THE ORION SEARCH ALGORITHM

ORION provides an efficient algorithm for keyword-based file search in MANET by combining application-layer query processing with the network layer process of route discovery. It combines application layer tasks with techniques known from the Ad Hoc On Demand Distance Vector (AODV) routing protocol [9] and the Simple Multicast and Broadcast protocol for MANET [6].

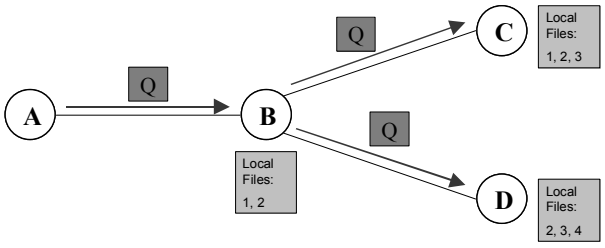

Figure 1. Node A floods a QUERY message with keywords matching to files 1 to 4

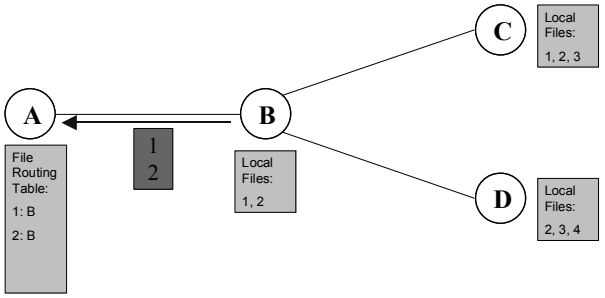

Figure 2. Node B sends a RESPONSE message with identifiers of local files

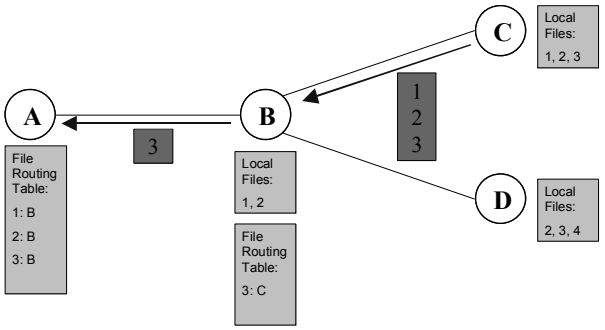

Figure 3. Node C sends a RESPONSE message with identifiers of local files, Node B filters and forwards RESPONSE of C

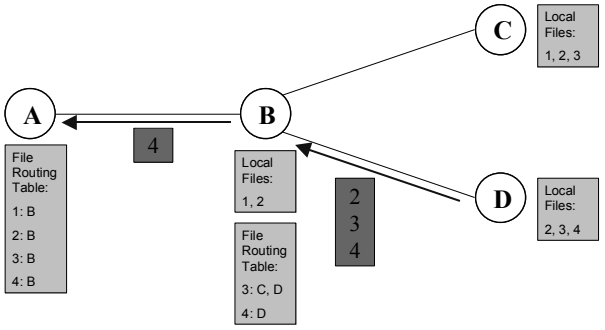

Figure 4. : Node D sends a RESPONSE message with identifiers of local files; Node B filters and forwards the RESPONSE message

To implement ORION, each mobile device maintains a local repository, consisting of a set of files stored in the local file system. ORION provides searching capabilities for all files in the repository. We assume that each file is associated with a unique file identifier. Additionally, ORION maintains two routing tables, a response routing table and a file routing table. Similar to the routing tables used by AODV, the response

routing table is used to store the node from which a query message has been received as next hop on the reverse path. Thus, a node is able to return responses to the enquiring node without explicit route discovery. The file routing table is a data structure that stores alternative next hops for file transfers based on the file identifier. ORION updates both the response routing table and the file routing table during query processing. The memory consumption for both routing tables is controlled by limiting the maximum size and applying the Least Recently Used (LRU) replacement policy.

For the query phase, ORION defines two types of messages. A QUERY message contains a query string, which consists of one or more keywords. A message of type RESPONSE contains unique identifiers of one or more files matching a query. To enable controlled message forwarding, each ORION message contains a field SRC, representing the unique identifier of the mobile device that generated the original message. Furthermore, a field SEQ contains a sequence number unique and strictly increasing for each mobile device. The rollover of SEQ numbers is handled as in [9]. When a mobile device relays a message as described below, SRC and SEQ are preserved. Thus, storing the largest SEQ entry received from each source device can prevent relaying a message more than once.

To illustrate the operation of the ORION search algorithm, we consider the four-node scenario shown in Figures 1 to 4. There, mobile nodes are shown as circles. Nodes are connected by a line when located in each others wireless transmission range. The rectangles near the nodes show parts of the local repository and the file routing table, respectively. Assume node A issues a query matching files 1, 2, 3 and 4 (Figure 1). The QUERY message is distributed by link-layer flooding, i.e., application data is piggybacked on a link-layer broadcast message. This technique is borrowed from the simple multicast and broadcast protocol for MANET. On its way through the network the QUERY message sets up reverse paths to node A in the response routing tables of all intermediate nodes. This technique is borrowed from the route discovery process in AODV, in which the flooded route request message sets up the reverse route for the route reply message.

After searching the local repository, node B sends a RESPONSE message containing the identifiers of files 1 and 2 to the next hop in node A's direction, i.e., directly to node A (see Figure 2). Additionally, node C sends a RESPONSE message with the identifiers of files 1, 2, and 3 in node A's direction, i.e., to node B. Before forwarding the message to node A, node B examines the contained result. File 3 is so far unknown to node B. Therefore, node C is stored as feasible next hop for this file in the file routing table. Node C is not stored as next hop for files 1 and 2, because node B stores these files itself. Therefore, node B will never request any of these two files from node C. Instead of relaying the complete RESPONSE message to node A, node B sends a reduced RESPONSE message to node A containing only the new file identifiers (see Figure 3). Similar to node C, node D answers the query with the identifiers of matching local files with a RESPONSE message to the next hop in the direction to node A, i.e., to node B. As before, node B stores node D's files in the file routing table and sends an own additional RESPONSE message to node A. Because node B has already forwarded responses for files 2 and 3, the new RESPONSE message only contains the identifier of file 4 (see Figure 4). Note that node B not only stores node C as next hop for file 3, but also node D, adding redundancy to the file routing table without additional transmission cost. After the query phase, node A may chose one of the four matching documents for download.

IV. THE ORION TRANSFER PROTOCOL

A. Basic Operation

As first building block, the ORION transfer protocol utilizes the routes given by the file and response routing tables for transmission of control- and data packets. Recall that the file routing table may store several redundant paths to copies of the same file. Due to changing network conditions, the sender of a file might change during a file transfer. Therefore, it is essential to keep the complete control over the transfer on the receiver side. Thus, opposed to TCP the ORION transfer protocol does not maintain an end-to-end semantic. For transfer, a file is split into several blocks of equal size. Since the maximum transfer unit of the mobile network is assumed to be equal between all neighboring nodes, the block size can be selected such that the data blocks fit into a single packet. The receiver sends a DATA_REQUEST message for one of the blocks along the path given by the file routing tables. Once the DATA_REQUEST reaches a node storing the file in the local repository, the node responds with a DATA_REPLY message, containing the requested block of the file. This message is routed back to the requesting node via the same path as the RESPONSE message in the query phase. After receiving the DATA_REPLY message, the receiver continues with a request for the next block until the complete file has been transferred. A scheduling mechanism described in Section IV.C prevents ORION from waiting indefinitely for lost packets.

To illustrate the operation of the file transfer mechanism, suppose node A in Figure 4 decides to download the file with the file identifier 3. The DATA_REQUEST message for the first part of the file is sent to the next hop in direction to the node storing file 3, i.e., node B. Node B cannot deliver the file itself, so it relays the request to the best-suited next hop towards a node storing file 3. From node B's point of view, node C is best suited, because in the query phase node C has responded first to the query message. Node C stores a copy of file 3 and, thus, sends a DATA_REPLY message containing the requested part of file 3 to the next hop in direction to node A as identified by the response routing table. Node B relays the DATA_REPLY message to node A. Subsequently, node A sends a DATA_REQUEST for the next block of file 3. This process continues until the complete file has been transferred to node A. In this (ideal) scenario, the file transfer implicitly uses the optimal route without any overhead for retransmissions due to route failures.

B. Maintenance of File Transfer Routes

ORION transfers control and data packets on the best-suited route chosen from a set of redundant routes. Selecting an alternative route provides an efficient mechanism to locally resolve link failures. Consider again the example in Figure 4.

Suppose the link between node B and node C fails during the file transfer, e.g., because node C moves out of node B's transmission range. As soon as B recognizes the link failure, it deletes node B in its routing tables and forwards subsequent DATA_REQUEST messages to the now best-suited next hop to a node possessing file 3, i.e. node D. Thus, the link failure can locally be resolved by node B without involving other nodes. Note that opposed to ORION, reactive MANET routing protocols as AODV and DSR recover from link failure by initiating a global route discovery, i.e., they will use global failure recovery. We will show in Section V.C that local failure recovery outperforms global recovery in terms of transmission overhead for a P2P file sharing application.

The timely recognition of link failures is necessary to avoid delays and unnecessary data transmissions. ORION uses feedback from the link layer as the second building block for an efficient file transfer. Feedback can be provided by link-layer notification if available, e.g., as in IEEE 802.11. Otherwise, packet receipt must be acknowledged by sending explicit application layer packets. In the case of a send failure on the link layer, the ORION application is notified and can immediately chose an alternative next hop as described above.

As a further feature, the route maintenance algorithm utilizes ROUTE_ERROR messages. These ROUTE_ERROR messages indicate that a node could not forward a DATA_REQUEST message, because there are no next hop entries for the requested file left in its file routing table. To illustrate the usage of ROUTE_ERROR messages, we refer again to Figure 4. Suppose that node B looses the link layer connections to both node C and node D and receives further DATA_REQUEST messages from node A. In that case, node B sends a ROUTE_ERROR message for file 3 to node A. Subsequently, node A deletes node B as next hop for file 3 in its files routing table. If node A's file routing table contains another next hop for file 3, subsequent DATA_REQUESTs are sent to this node. Otherwise node A can either cancel the file transfer or start a special search phase called re-query. Opposed to an ordinary query, a re-query does not search for keywords, but for a file identifier. The messages transmitted during a re-query are equal to the messages during an ordinary query, thus updating the routing tables for the specific file.

C. Packet Scheduling and Loss Recovery

Recall that the ORION transfer protocol as described in Section IV.A will not continue a file transfer, if a DATA_REQUEST or DATA_REPLY message is lost. Thus, the ORION transfer protocol incorporates a packet scheduling and loss-recovery mechanism as third building block. In the remainder of this paper, we demonstrate the general applicability of the file transfer protocol and do not elaborate on performance and fairness optimization using sophisticated flow control and congestion avoidance as for example provided by the transmission control protocol, TCP [2]. Recall that TCP tries to transmit data packets with a small delay jitter as a continuous stream. In a file sharing application, it does not matter in which order the blocks of a file are received, as long as each block is received at least once. Therefore, ORION maintains a list of pending blocks, which is initialized with all blocks of a particular file. The blocks are requested in a round-robin fashion, until they are successfully transmitted. To determine the minimal request rate, the receiver keeps track of the average round trip time, i.e., the time elapsed between sending a DATA_REQUEST and receiving the corresponding DATA_REPLY. The time between two successive requests is equal to the average round trip time, unless a DATA_REPLY is received before. That is, if a packet is lost or delayed, the next packet will be requested at most one round trip time later. Due to the round-robin selection of blocks, the ORION transfer protocol re-requests a lost or delayed block after all other pending blocks have been requested. Therefore, a delayed block will reach the receiver with high probability before it is requested a second time.

V. Performance Results

A. Simulation Environment

In this section, we compare the performance of both the ORION search algorithm and the ORION transfer protocol with an off-the-shelf approach using a P2P system for the wired internet, TCP and a MANET routing protocol. Performance results were obtained using the Network Simulator ns-2 [3]. We used an IEEE 802.11 standard MAC layer [4], and a standard physical layer using two-ray ground propagation as radio propagation model. With a transmission power of 17.6125 mW, the simulated mobile nodes provide a transmission range of approximately 115m.

We assume that N_{Nodes} mobile devices move in an area of 1000 m × 1000 m according to the random waypoint mobility model, which is commonly used to model the movement of individual pedestrians. The speed of the device is chosen uniformly at random from $[0, s_{\max}]$. When a device reaches a randomly chosen destination, it pauses for a fixed amount of time T_{hold}, before it continues to move to the next destination.

TABLE I. Default parameters.

Parameter	Value
Transmission Range	115 m
Number of Nodes N_{Nodes}	40
Simulation Area	1000 m × 1000 m
Maximum Speed s_{max}	2 m/s
Rest Time T_{hold}	50 s
TCP Timeout T_{TCP}	60 s

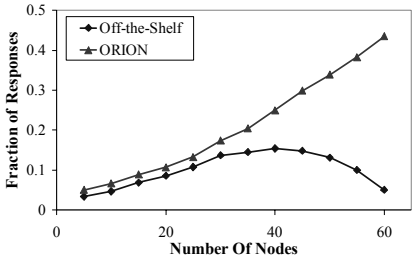

Figure 5. Search accuracy vs. size

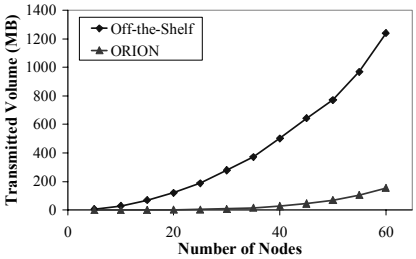

Figure 6. Protocol overhead vs. size

If not stated otherwise, all fixed parameters are set according to Table 1. We calculated 99% confidence intervals for the simulation results and indicate them as bars for each data point in a performance curve.

B. Search Performance

To illustrate the effectiveness of ORION's search algorithm, we compare its performance with an off-the-shelf approach utilizing a P2P system for the wireline Internet, TCP, and a state-of-the-art MANET routing protocol. Since almost all known P2P file sharing systems use static connections to build an overlay network we consider Gnutella [7] as example for such P2P systems. For the off-the-shelf approach, we employ TCP-SACK as transport layer protocol on top of the MANET routing protocol Dynamic Source Routing (DSR, [5]). We label corresponding curves as ORION and Off-the-Shelf, respectively.

An important performance measure from the user's point of view is the quality of the results received in response to a sent query. Therefore, we consider search accuracy, i.e., the fraction of received unique files in relation to the number of all files matching a query. In Figure 5, we plot search accuracy as a function of number of nodes participating in the mobile file sharing application. We find that search accuracy for both applications increase with the number of nodes. This is clearly due to the increasing connectivity. However, the off-the-shelf approach has searching performance worse than ORION and also shows decreasing search accuracy for more than 40 mobile nodes.

To evaluate the overhead generated by both approaches, we investigate the accumulated volume of transmitted messages for an increasing number of nodes. Figure 6 shows, that ORION outperforms the off-the-shelf approach by more than one order of magnitude for a MANET with a large number of nodes. we conclude that P2P file sharing systems based on a static application-layer connections are not applicable even for environments with few nodes, while ORION easily scales to large scenarios.

Figure 7 gives insight in the composition of messages contributing to the overall message volume generated by both approaches. Message volume in the off-the-shelf approach is dominated by control traffic from MAC, routing and transport layers. Only 5% of the overall traffic is used for exchange of query and response messages (payload), which should be the main purpose of the search algorithm in a P2P application. In contrast, the fraction of this type of traffic exceeds 79% of the total message volume for ORION. Since the amount of payload is similar in both approaches, this experiment clearly shows that the main deficiency of the off-the-shelf approach is not the inefficient query mechanism but the overhead of maintaining static overlay connections. Therefore we argue that the results obtained with the off-the-shelf-approach are also valid for any P2P file sharing systems using static overlay connections, not only for Gnutella. We conclude from Figure 7 that the off-the-shelf approach fails to make efficient use of network bandwidth, while ORION utilizes network resources effectively.

C. Transfer Performance

For evaluating the performance of the ORION transfer protocol, we compare it with an off-the-shelf file transfer approach using TCP as transport layer protocol on top of DSR and AODV, respectively. We found that TCP throughput is roughly the same for DSR and AODV. Therefore, in this section we present the performance results for TCP with DSR and omit corresponding results for AODV. We refer to the considered transfer protocols as ORION and Off-the-Shelf, respectively. For each point in the performance curves, we performed 20,000 queries and successive file transfers. Each transferred file has a size of 3 MB, reflecting the average size of an MP3 file. The degree of replication is 0.1 in all experiments, i.e., 10% of the participating nodes store copies of a given document.

In a P2P file sharing system, an important feature for achieving user satisfaction is the number of successful file transfers. We analyze this performance measure in Figure 8. For the off-the-shelf approach, a download is considered as "failed", if none of the nodes responding to the original query remains reachable. In the case of ORION, a download is considered as "failed" when the inquiring node runs out of alternative paths after a single re-query. Note that ORION is able to implicitly discover file holders that are unreachable when issuing the original query. Figure 8 shows that due to low connectivity in systems with a low number of nodes, both approaches are able to complete only a small number of file transfers. For a growing number of nodes, ORION is able to complete up to 40% more transfers, based on the file-oriented re-query process. We conclude that ORION can provide high user satisfaction by completing a large fraction of file transfers.

One of the most important performance measures for data-intensive applications running on mobile devices is the transmitted data volume, due to scarce bandwidth and energy resources. In a P2P file sharing application, a trivial lower bound for the volume generated by a file transfer is the size of the file. This bound holds under the assumption that the

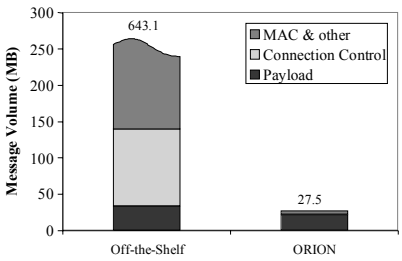

Figure 7. Breakdown to message types

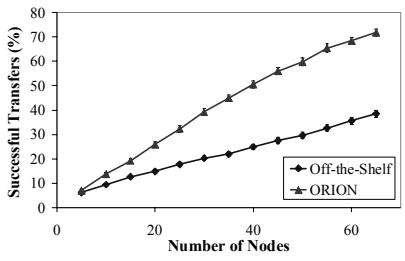

Figure 8. Successful transfers vs. size

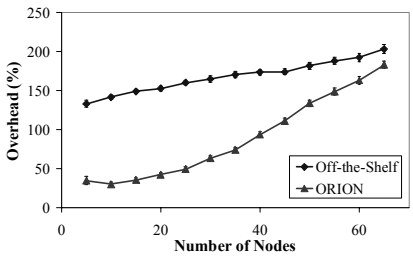

Figure 9. Overhead vs. size

network route used for the transfer spans only a single hop and that there is no overhead for packet headers, flow control and packet retransmissions. However, the transmission might experience significant overhead due to network routes spanning more than one hop as well as due to retransmissions of lost or delayed packets.

We investigate the average overhead for a successful file transmission in percent of the total file size of 3 MB. Figure 9 plots the overhead as a function of the number of nodes. For few nodes, in most cases files are transferred between direct neighboring nodes. In this case, most overhead is control overhead. We find that the off-the-shelf approach transmits more than twice the file size over the wireless medium. As shown by an analysis of the ns-2 trace files, the main reason for this are retransmissions of packets delayed by the route maintenance of DSR, as well as longer transfer routes. Route length increases if a direct connection between the sending and receiving node fails, while a longer route via an intermediate node exists. Recall that in this case, DSR will recover from route failure globally, using the longer route. The scheduling algorithm of ORION reduces the amount of retransmissions of delayed packets. Furthermore, in case of route failure, ORION will resume the transmission from another nearby node using the file routing table. These features lead to a total overhead below 50% for few nodes. For a growing number of nodes, overhead increases for both the off-the-shelf approach and ORION, as the average length of a network route increases. However, ORION stays superior to the off-the-shelf for the same reasons. From Figure 9 we conclude that ORION provides a file transfer mechanism that is highly efficient in bandwidth usage.

In further experiments, we compared the throughput of the ORION transfer protocol to the off-the-shelf approach. The experiments show that the throughput is comparable for both approaches. Note that a congestion control mechanism for the ORION transfer protocol is subject to future work, thus we did not consider cross traffic in our experiments. Due to space limitations, we do not discuss the results in detail.

VI. CONCLUSION

In this paper, we presented the Optimized Routing Independent Overlay Network (ORION), a special-purpose approach for P2P file sharing tailored to MANET. ORION is completely implemented on the application layer and does not depend on support of a MANET routing protocol. As building blocks, ORION comprises of an algorithm for construction and maintenance of an application-layer overlay network that enables routing of all types of messages required to operate a P2P file sharing system, i.e., queries, responses, and file transmissions. It combines application-layer query processing and overlay network construction with the network layer process of route discovery and transparently aggregates redundant transfer routes on a per-file basis. The ORION transfer protocol enables efficient file transfers on top of the overlay connections established by the search algorithm.

In a detailed ns-2 simulation study, we illustrated the performance gains of ORION compared to an off-the-shelf approach based on a P2P file sharing system for the wireline Internet, TCP and a MANET routing protocol. We found that ORION significantly increases search accuracy and reduces overhead for searching. Furthermore, ORION enables more reliable file transfers with lower overhead compared to the off-the-shelf approach.

In future work, we will combine the content based routing facilities provided by ORION with reactive MANET routing protocols in order to use synergies in mechanisms and data structures. Closely integrated with the functionality of the network layer, the key concepts of ORION can provide a general-purpose foundation not only for P2P file sharing but also for many other MANET based P2P applications. Furthermore, a congestion control mechanism for the ORION transfer protocol will be developed to guarantee fairness with respect to other transfers.

REFERENCES

[1] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt, Improving Data Access in P2P Systems, IEEE Internet Computing 6(1), 58-67, 2002.

[2] M. Allman, V. Paxson, and W. Stevens, TCP Congestion Control, IETF RFC 2581, 1999.

[3] K. Fall and K. Varadhan (editors), The ns-2 manual, Technical Report, The VINT Project, UC Berkeley, LBL, and Xerox PARC, 2002.

[4] IEEE Computer Society LAN MAN Standards Committee, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-1997, New York, NY, 1997.

[5] D. Johnson, D. Maltz, Y. Hu, and J. Jetcheva, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR), http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, IETF Internet Draft (work in progress), February 2002.

[6] J. Jetcheva, Y. Hu, D. Maltz, and D. Johnson, A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks, http://www.ietf.org/proceedings/01dec/I-D/draft-ietf-manet-simple-mbcast-01.txt, IETF Internet Draft (work in progress), July 2001.

[7] T. Klingberg and R. Manfredi, Gnutella 0.6, Draft, 2002. http://groups.yahoo.com/group/the_gdf/files/Development/GnutellaProtocol-v0.6-200206draft.txt

[8] C. Lindemann and O. Waldhorst, A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications. Proc. 2nd IEEE Conf. on Peer-to-Peer Computing (P2P 2002), Linköping, Sweden, 71-83, 2002.

[9] C. Perkins, E. Royer, and S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, IETF RFC 3561, 2003.

[10] M. Papadopouli and H. Schulzrinne, Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination among Mobile Devices, Proc. ACM Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC 2001), Long Beach, CA, 117-127, 2001.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, A Scalable Content-Addressable Network, Proc. ACM SIGCOMM Conf. 2001, San Diego, CA, 161-172, 2001.

[12] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and Balakrishnan, H, Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, Proc. ACM SIGCOMM Conf. 2001, San Diego, CA, 149-160, 2001.