Abstract

Peer-to-peer (P2P) networks and mobile ad-hoc networks (MANET) share many similarities, such as decentralisation, dynamic topology and auto-configuration. Both types of networks also share a common goal: the discovery of nodes, according to specific criteria, along with the routing of messages between such nodes. Despite this, existing P2P systems, designed for the Internet, do not exploit the synergies between P2P networks and MANETs since the P2P overlay network is unaware of the topology of the underlying physical network. This can lead to inefficient paths between neighbours in the overlay and unnecessary network overhead due to the duplicate functionality implemented at different layers in the network protocol stack.

This paper proposes a P2P middleware solution to facilitate the development of P2P applications optimised for use in MANETs. To achieve this, the process of searching for P2P application content is combined with the route discovery process of the MANET routing protocol, Hybrid Wireless Mesh Protocol (HWMP). An implementation of the proposed middleware is shown to improve performance and reduce the network resource use when compared to a traditional P2P system layered on top of a MANET.

1 Introduction

The rapid growth of wireless communication and demand for mobility have led to the appearance of Mobile Ad-hoc Networks (MANETs). These fully decentralised networks comprise a set of autonomous wireless mobile nodes which are not dependent on existing fixed communication infrastructure to operate. These networks have been proposed for a variety of scenarios [1] where users cannot rely on an existing network infrastructure: it may be unreliable, too expensive or simply inexistent. The lack of adequate communications infrastructure is frequently the case in remote communities and developing countries where MANETs can provide an attractive solution to enable network connectivity.

Another attractive network concept is that of Peer-to-Peer (P2P) networks which allow direct sharing of resources (e.g., CPU, bandwidth, storage) among a large number of users in a decentralised manner [2]. These networks can serve many purposes, from digital content distribution, to real-time communications and collaborative applications. A P2P network is an application layer overlay network, formed on top of the underlying physical network, which interconnects its various participants, i.e., the peers. In completely decentralised P2P networks, such as Gnutella [3], all peers perform the same functions and are capable of sending and replying to request messages sent by any other peer, without requiring a central element to manage and coordinate network interactions. P2P technologies can therefore provide a powerful mechanism to enable information exchange and resource discovery in the scenarios typically envisioned for MANETs.

The synergies between these two networking concepts provide an interesting research topic [4]. Both MANETs and P2P networks follow a P2P paradigm characterised by the lack of a central node or peer acting as a managing server, all participants must therefore collaborate in order for the whole system to work. A key issue in both networks is the process of discovering the requested
data or route efficiently in a decentralised manner. MANETs and P2P networks also have a highly dynamic topology: nodes are constantly moving in and out of radio coverage of neighbours while peers are continuously joining or leaving the network.

Although very much alike, the integration of both these concepts presents some important challenges. P2P networks aim to provide resource sharing on top of existing reliable communication infrastructures, while MANETs focus on providing multi-hop wireless connectivity where no infrastructure exists or is inadequate. P2P application networks operate at the application layer of the OSI reference model while MANETs operate at the network layer or data link layer. Routing functionality is therefore present in both layers. On the one hand, overlay routing is used by search queries to find a peer containing the desired information. On the other hand, the MANET routing protocol is used to discover the route to any given node. Another difference lies in the fact that nodes in a MANET are normally mobile, constrained in computational power, energy and communication bandwidth which is usually not an issue for peers connected via wired infrastructure to Internet wide P2P applications. Furthermore, most existing P2P systems form an overlay network that is unaware of the underlying physical network topology. Neighbouring nodes in the P2P overlay can in fact be located on opposite sides of the network which can lead to long and inefficient routes therefore generating unnecessary traffic. This represents a significant problem for MANETs due to their limited resources and reliability.

Preliminary theoretical analyses [4] and practical simulations [5] have shown that merely applying existing P2P systems designed for the Internet on top of MANETs can lead to poor performance and unnecessary network traffic. Such results illustrate the need for P2P systems that are specifically designed to take into account the unique properties of MANETs on top of which they are deployed, including their physical topology. This can be achieved by following a cross-layer approach in which the P2P application layer and MANET routing layer are integrated or interact closely with each other in order to reduce duplicate functionality, optimise network resource usage and path selection [6].

In order to facilitate the development of a wide range of new P2P applications optimised for use in an IEEE 802.11 MANET environment and the adaptation of existing ones from the Internet, we propose a novel P2P middleware approach that exploits the route discovery functionality and information provided by the underlying MANET routing layer to improve the application layer P2P search performance. The proposed P2P middleware is aware of the underlying network topology and maps logical neighbours in the P2P overlay to physical neighbours in the MANET. Furthermore, the reactive MANET routing protocol is extended to combine the process of searching for information in the P2P network with the process of discovering the network routes to the peers containing such information.

This paper is structured according to the following sections. The next section, Section 2, provides a description of the architecture of the proposed P2P middleware which is tailored specifically for use in a MANET scenario. Section 3 offers an overview of the implementation of the middleware, developed for the GNU/Linux operating system. Section 4 presents experimental performance results obtained for the middleware implementation using a MANET testbed. Finally, Section 5 concludes this paper.

2 Architecture

In designing the architecture of the proposed P2P middleware, several assumptions and requirements were taken into account. We assume MANET deployment scenarios, which, due to their higher mobility, churn and/or network partitions, are unsuitable for use with structured P2P systems which are characterised by a higher overlay maintenance overhead in the face of dynamic network conditions [7]. We try to reduce broadcast network traffic where possible and therefore propose the integration of the MANET route request messages with P2P search request messages. Furthermore, we try to reduce the mismatch between the overlay network and the physical network. The architecture of the middleware can be divided into the network architecture and the peer, or system, architecture. A comprehensive description of both architectures follows.
2.1 Network Architecture

The proposed solution combines the use of an unstructured P2P overlay, because of its lower overlay maintenance overhead in the presence of frequently changing network conditions and support for a more comprehensive set of search queries when compared to a structured P2P overlay; with a reactive MANET routing protocol, on account of its greater efficiency in terms of control overhead and increased suitability for networks with moderate mobility when compared to other flat ad-hoc routing protocols. The Hybrid Wireless Mesh Protocol (HWMP) was chosen as the MANET routing protocol due to its inclusion in the IEEE 802.11s draft, a mesh networking amendment to the current IEEE 802.11 standard [8]. IEEE 802.11s promises to become the de facto standard for the creation of multi-hop wireless mesh networks and MANETs. Although HWMP is a hybrid routing protocol, incorporating both reactive and proactive components, only its reactive component is considered. HWMP’s reactive component is based on a modified version of the well known AODV [9] routing protocol, adapted for layer 2 address-based routing and radio-aware link metrics.

Given the parallel between unstructured P2P overlay networks and MANETs employing reactive routing protocols, the proposed solution avoids duplicating the discovery and routing functionality at different layers in the network stack by integrating the process of searching for information in the P2P overlay with the discovery of the network routes in the underlying MANET. Rather than use a P2P overlay network consisting of a set of persistent connections setup between possibly distant peers in order to route application layer search requests and replies, the HWMP’s routing mechanisms are used instead. This is achieved by extending the HWMP path request (PREQ) and path reply (PREP) messages in order to transport, respectively, application layer search requests and search replies. Therefore, one of the primary services provided by the middleware consists in allowing P2P applications to submit new search requests to the network routing layer and notifying said applications of received search requests.

This integrated approach offers several advantages over the use of two distinct networks. The MANET is exclusively responsible for the maintenance of the network topology so that any changes introduced by network participants, such as nodes failing or moving in and out of range of other nodes, are immediately reflected in the P2P overlay. The P2P overlay does not suffer from a high path stretch, defined as the ratio of physical hops traversed on the path between two peers in the overlay to the number of physical hops on the direct unicast path between those peers, because overlay messages are forwarded along the latter. Thus, no mismatch exists between the P2P overlay network topology and the underlying MANET topology. Furthermore, traffic generated for overlay maintenance purposes is reduced, as peers do not need to maintain persistent connections to other peers in the network. Finally, the packet overhead for application layer searching and the subsequent route discovery is lower than if both tasks were performed separately.

To illustrate the operation of the proposed solution, consider the multi-hop MANET shown in Figure 1. Let \( P = \{A, B, C, D, E\} \) be the set of peers in the network and \( S = \{1, 2, 3, 4\} \) the set of searchable documents stored in the network. Multiple copies of the same document may be available from different peers. Furthermore, consider an application at peer A which issues a query matching documents 3 and 4. The application starts by informing the middleware of the search query. An HWMP PREQ frame containing a search request is generated by the middleware and broadcast using HWMP to the set of physical neighbours.

![Figure 1: Searching for documents in a MANET by leveraging HWMP route discovery](image-url)
i.e. the peers which are within radio transmission range: B and C. Each neighbour peer, upon receiving the PREQ frame, informs the middleware of the received search request before any further forwarding action is taken. Also, HWMP stores peer A’s address as the next hop on the reverse path to the requesting peer. The middleware searches its local document repository for any documents matching the query contained in the search request. Since neither peer has a matching document, the middleware informs HWMP of its forwarding decision and the PREQ frame is again broadcast to all neighbouring peers. Propagation of the search request throughout the network is limited by the time-to-live (TTL) value defined for the enclosing PREQ frame and duplicate search requests are discarded at each peer. Finally, the PREQ containing the search request reaches peer D. HWMP stores peer B’s address as the next hop on the reverse path to the requesting peer, peer A. Since matching documents are available at peer D, the discovered reverse path is used to unicast a HWMP PREP frame containing a search reply back to peer A. This search reply includes information necessary to contact peer D, such as its network address and transport layer ports, along with the actual search results. It is important to note that the size of the search results which can be included in this search reply is limited by the maximum transmission unit (MTU) of the underlying IEEE 802.11 network interface, thus, the requesting peer may need to use an alternate mechanism to obtain the complete result set. As the PREP is unicast to peer B and subsequently to peer A, the forward path between both peers is discovered. The behaviour of peer E is analogous as it too will generate a PREP containing a search reply which will then be unicast back to peer A through peer C.

Peer A, upon receiving both PREPs containing the search replies, knows of one source peer, peer D, for document 3 and two source peers, peer D and peer E, for document 4. Additionally, as path discovery was executed in parallel to the searching process, valid bidirectional network paths are now available for reaching peer D and peer E from peer A and vice-versa. The application can use the information contained in the search replies to contact the peers and obtain the requested documents. Further communication between peers is achieved by using the discovered unicast paths along with an appropriate transport protocol. If subsequent changes to the network topology cause said paths to break, HWMP’s path failure notification and discovery mechanisms are used to discover alternate paths.

In the case of document 4, which is available from multiple peers, the application is faced with an additional problem: deciding which peer it should contact to transfer the document from. HWMP uses a link metric when building and selecting network paths [10]. Upon reaching their destination, PREQ and PREP frames contain the cumulative value of the link metric calculated at each hop along traversed path. This value is stored along with the remaining forwarding information for each path. IEEE 802.11s defines a default link metric to ensure compatibility, the airtime link metric, which defines the cost, i.e. the amount of network resources consumed, to transmit a frame over a given link [11]. The proposed solution uses the path metric for paths discovered by HWMP as a criteria for selecting a source peer for a given document.

Consider the network paths and corresponding link metrics shown in Figure 2, discovered as a result of the search process described in the aforementioned example. Furthermore, consider the use of a bi-directional airtime link metric, that is, the cost of transmitting a frame from peer X to peer Y is equal to the cost of transmitting the same frame from peer Y to peer X. Peer A, given the availability of two source peers for document 4, would consult HWMP’s forwarding information table to obtain the path metric for both destinations: $8 + 3 = 11$ in the case of peer D and $10 + 5 = 15$ in the case of peer E. Due to the lower cost of the path from peer

<table>
<thead>
<tr>
<th>Peer</th>
<th>Path metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>11</td>
</tr>
<tr>
<td>E</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 2: Source peer selection using HWMP path metrics
A to peer D, the application would opt to transfer document 4 from peer D. The middleware thus provides P2P applications with an optimised, network-aware, source peer selection service when multiple sources are available for the same content.

2.2 Peer Architecture

The P2P middleware provides the functionality required to establish the network architecture described in the previous section. An overview of the peer architecture is presented in Figure 3, where the middleware is introduced between the application and the existing operating system services. The middleware is further divided into a set of interconnected components, each with a well defined set of responsibilities:

- **P2P-HWMP** - Provides an interface for use by the remaining middleware components, namely the query routing service, in order to send, and be notified of, search requests and search replies contained in HWMP frames. P2P-HWMP is responsible for encapsulating upper layer search requests and replies in the appropriate HWMP PREQ and PREP frames, notifying the query routing service of received requests and replies, and acting on query forwarding decisions provided by the query routing service.

- **Storage service** - Provides a local repository for storing and indexing application content in addition to being responsible for resolving queries, such as those received in search requests, to determine whether any matching content is available.

- **Query routing service** - Responsible for generating new search requests containing application supplied search queries, deciding on how received search requests should be forwarded by the underlying routing protocol, HWMP, and generating search replies in response to received search requests.

- **Service registration service** - Manages the registration of services provided by P2P applications and the middleware, allowing them to be discovered by other peers in the network.

- **Peer selection service** - Performs source peer selection using HWMP path metrics and maintains a list of known peers in the network.

- **Peer management service** - Provides configuration information for other middleware services and enables remote monitoring of the middleware.

- **Middleware communication service** - A P2P application interacts with the middleware through an API provided by the middleware communication service. This service is responsible for dispatching application requests to the appropriate middleware service and for enabling generic middleware to middleware communication, beyond that which is provided by the query routing service.

Other than these components, HWMP was extended to support middleware supplied information and interaction with the middleware.

To exemplify the interactions between the various components, consider a scenario similar to the one presented in Section 2.1, where a P2P application wishes to search the network for documents matching a given query string. The application starts by calling the search function provided by the middleware API and passing an appropriate query string, as illustrated in Figure 4. The middleware communication service dispatches this request to the corresponding service, the query routing service, which will generate a new search request containing the query string. The newly generated search request is sent to P2P-HWMP along with the indication of the TTL to use for the enclosing PREQ frame and the address to which this frame should be sent, typically the broadcast address. P2P-HWMP then uses HWMP to construct...
and send a PREQ frame containing the search request.

A neighbour peer, upon receiving a PREQ frame, will contact P2P-HWMP for further processing, as illustrated in Figure 5. P2P-HWMP extracts the search request from the PREQ frame and notifies the query routing service, which in turn contacts the storage service in order to determine if any locally stored documents match the query string contained in the search request. If no documents match the received query, the query routing service notifies P2P-HWMP of its decision to continue forwarding the search request, along with the address to which the search request should be forwarded. Otherwise, if any documents match the received query, the query routing service generates a new search reply containing the peer’s contact information and, depending on their size, the actual search results. The search reply is sent to P2P-HWMP which will in turn generate and send a PREP frame using HWMP, containing the search reply, in response to the received PREQ. This PREP frame is unicast back to the requesting peer. Finally, once the PREP frame is received at the requesting peer, the enclosed search reply travels up to the middleware communication service which ultimately returns the search results to the P2P application.

Figure 4: Interactions between middleware components at the requesting peer

Figure 5: Interactions between middleware components at the receiving peer

3 Implementation

An implementation of the middleware described in the previous section was developed for the GNU/Linux operating system. Current versions of the Linux kernel, 2.6.26 and later, include open80211s [12], the first and foremost open source implementation of the IEEE 802.11s draft, namely of its default routing protocol, HWMP, and link metric, the airtime link metric. open80211s is part of the new Linux IEEE 802.11 stack, mac80211, and allows nodes with compatible hardware to establish infrastructure-less wireless networks such as Wireless Mesh Networks (WMN) and MANETs. A modified version of this HWMP implementation, included with version 2.6.29 of the Linux kernel, serves as a basis for the developed middleware.

As a result of the in-kernel nature of open80211s, the middleware is comprised of both kernelspace and userspace components. P2P-HWMP, which is tightly coupled to HWMP, is implemented as a loadable kernel module (LKM) while the remaining middleware components are implemented in userspace. Kernel-space components are therefore limited to those which must interact directly with HWMP, i.e., P2P-HWMP, so as to limit the amount of code running with elevated privileges as well as to reduce implementation complexity and avoid compromising the stability of the Linux kernel. P2P-HWMP invokes functions exported
by HWMP and communicates with the userspace middleware components via netlink sockets, an asynchronous inter-process communication mechanism similar to UNIX datagram sockets. Several hooks were added to HWMP to call P2P-HWMP when specific information elements are found within PREQ and PREP frames.

The majority of the userspace middleware components are implemented by a daemon which runs as a background process, independent of the P2P applications. This approach was chosen over the use of a library, called by each application during its execution, in order to enable persistent functionality, such as the discoverability of services or remote monitoring, despite the lack of currently running applications. Furthermore, the use of a single middleware instance reduces the memory footprint required by applications and improves their startup time. Given that the storage service stores and manages all shared application content, an application need not be running in order for its content to be searchable and retrievable.

Lastly, P2P applications use the middleware API in order to interact with the middleware. This API corresponds to a thin library which employs an appropriate inter-process communication (IPC) mechanism to contact the middleware daemon. The API hides the fact that the remaining userspace middleware components are running in a separate process as well as the protocol used to interact with the middleware daemon. Figure 6 illustrates the software stack at each peer. The implementation of the middleware based solution consists of four distinct parts: the modification of the Linux HWMP implementation, the P2P-HWMP kernel module, the middleware daemon and the API library.

4 Testing and Evaluation

In order to validate the implemented middleware solution and evaluate its performance, a set of functional and performance tests were conducted using a MANET testbed. The testbed consisted of six nodes equipped with IEEE 802.11 network interfaces forming a multi-hop MANET using IEEE 802.11s and its default routing protocol, HWMP. The primary focus of these tests was the search functionality offered by the middleware, and the resulting query routing, as this, along with source peer selection, constitutes the core functionality of the middleware. In the context of the performance tests, the middleware was also compared to a traditional layered solution, using Gnutella, which was not designed for use in a MANET environment.

The metrics used in for evaluating the middleware’s performance consisted of the query forwarding delay, query success rate, overall traffic and query processing time. The query forwarding delay represents the time elapsed between the transmission of a received search request, by P2P-HWMP, to the query routing service and the reception of the forwarding decision for that search request. This metric is used to evaluate the overhead introduced by taking forwarding decisions regarding a search request in userspace. The query success rate is defined as the fraction of queries for which at least one reply was received over the number of queries sent. The query success rate only includes queries for which at least one matching document exists at another peer in the network. The overall traffic, both in terms of total size and number of messages, refers to the network traffic which is generated as a direct consequence of a search request and includes the overhead introduced by the IEEE 802.11 MAC, HWMP path discovery, ARP address resolution, transport and application-layer protocols. Lastly, the query processing time refers to the wall clock time spent processing a given search request at an intermediate peer, i.e, the time elapsed between re-
ceiving a PREQ frame and forwarding it to the next hop. This processing time is divided according to the middleware components involved, namely P2P-HWMP, netlink communication, the query routing service and the storage service. Although similar to the query forwarding delay metric, the query processing time is more encompassing and provides insight into which middleware components have the greatest impact on the time required for a search request to traverse a peer.

For the purpose of evaluating the performance the middleware, a test scenario using a simple file-sharing application was run at each peer. This application uses the middleware API to create a document repository and creates a file-backed document for each file in a specified directory. These file-backed documents, which represent the metadata associated with a file in the peer’s filesystem, are then added to the repository, which is marked as shared, enabling other peers to discover the files contained in the directory. Each peer shares a directory containing a few dozen files. To test the correct operation of the middleware search functionality, a peer sends a search request for a specific file shared by another peer in the network such that at least two intermediate peers exist between the requesting peer and the replying peer. This constraint is used to exercise query forwarding over multiple hops. The expected result consists in the reception of a search reply for that request. This scenario was used to obtain experimental results for the aforementioned metrics. No background traffic was present in the MANET during each measurement, with the exception of the traffic required by the IEEE 802.11 MAC, namely periodic beacon transmissions announcing the presence of each node.

Figure 7 presents the results of measuring the query processing time at a given peer for a received query matching a stored document. Both P2P-HWMP and netlink communication represent a negligible contribution, 0.78 ms, to the overall processing time. Both these components are executed in a kernel interrupt handler and therefore do not suffer from scheduling delays. Furthermore, these components carry out only a limited set of operations on the received search request. The bulk of the processing time, 8.6 ms, is split between the query routing service, where the search request is parsed, and the storage service, where the query is resolved by consulting a document repository. These services represent an opportunity for further optimisation with regard to the code path followed in order to forward a search request. In spite of this, the time taken by a query to traverse the middleware at a given peer, 9.38 ms, is in line with that estimated for the one-hop packet traversal time for traditional MANET routing protocols. For AODV, the estimated packet traversal time, including the time required to transmit the packet to the next hop and any queuing delays, is 40 ms [9].

A comparison between the middleware and an off-the-shelf solution consisting of Gnutella, TCP/IP and HWMP was used to analyse the efficiency of the implemented middleware solution with respect to the overall traffic generated as a result of a single query, i.e., the volume of unicast and broadcast traffic generated, both in terms of the number of frames as well as the total size. gtk-gnutella was chosen as the Gnutella client and installed on every node in the testbed alongside the middleware. gtk-gnutella is a popular Gnutella client for Unix-like operating systems which implements the latest version of the Gnutella protocol, version 0.6. gtk-gnutella was configured in ultrapeer mode, without any leaf nodes, in order to reflect the query flooding nature of Gnutella and the lack of node hierarchy inherent in a MANET.

One node was configured to share a given file using both gtk-gnutella and the middleware-based P2P file-sharing application. At another node, each application was then used to send a query for this file. An additional node was used to capture all traffic exchanged between the nodes participating in the MANET, during a 30 second interval, for
a query sent according to three scenarios: using the file-sharing application developed with the middleware API, henceforth known only as middleware, using gtk-gnutella with a previously established overlay, henceforth known as Gnutella warm, and using gtk-gnutella while establishing the overlay during the measurement interval, henceforth known as Gnutella cold.

The overall traffic generated as a result of the query, in terms of total size of the transmitted IEEE 802.11 frames, is displayed in Figure 8 for the three scenarios. The composition of the total size is also shown, where each transmitted frame is classified according to its payload: IEEE 802.11 MAC management frames, HWMP path selection frames (excluding those with middleware payload), TCP, the various Gnutella protocol messages (query, query hit, ping, pong), frames containing middleware search requests or search replies and finally other frames used, for example, for ARP address resolution requests and responses. In the case of the middleware scenario, only two types of frames are generated: IEEE 802.11 MAC management frames and HWMP path selection frames containing search requests or search replies. The middleware scenario represents a 74% decrease in traffic when compared to the Gnutella warm scenario and an 85% decrease when compared to the Gnutella cold scenario. The overhead introduced by Gnutella is mainly due to the use of Gnutella Ping and Pong messages for discovering other peers in the overlay and determining the status of connections to neighbouring peers. Other factors contributing to the overhead include the successive HWMP route discoveries triggered in order to maintain persistent TCP connections between overlay neighbours, as well as, the overhead introduced by TCP itself including any packet re-transmissions.

Additionally, the number of messages broadcast at the link-layer for each scenario is a factor in determining the efficiency of a given solution, as broadcast traffic has a greater impact on network performance than unicast traffic. Figure 9 shows the number of messages broadcast for the three scenarios, excluding IEEE 802.11 MAC beacon frames, as these are always present, with the same frequency, regardless of the scenario. In the case of Gnutella, broadcasting a query using the overlay results in multiple messages unicast at the link-layer, therefore, such virtual broadcasts are omitted. Despite this, the number of messages broadcast for the Gnutella warm and Gnutella cold scenarios represent an increase of 2040% and 2900%, respectively, over the middleware scenario. While the middleware executes a single HWMP path discovery process to propagate the query to all peers and receive the corresponding reply, Gnutella uses static TCP connections between overlay peers. Keep-alive traffic, in other words, ping and pong messages, are constantly exchanged over these connections. Since the paths discovered by HWMP expire after 5s, each Gnutella peer will trigger a path discovery to each of its current overlay neighbours in order to maintain the corresponding overlay connection. The PREQ frames which are broadcast as a result of successive path discovery processes contribute to Gnutella’s poor result.
5 Conclusion

In this paper we proposed a P2P middleware solution to facilitate the development of P2P applications optimised for use in MANETs. This solution avoids duplicating the discovery and routing functionality at different layers in the network stack by integrating the process of searching for application content with the discovery of the network routes in the underlying MANET. Rather than use a P2P overlay network consisting of a set of persistent connections setup between possibly distant peers in order to route application layer search requests and replies, the MANET routing mechanisms are used instead. Furthermore, when multiple sources are available for the same content, the middleware chooses a peer based on the cost of the path to that peer, as given by HWMP’s path metric.

An implementation of the proposed P2P middleware was developed for the GNU/Linux operating system and based on the HWMP and airtime link metric implementations included with the Linux kernel IEEE 802.11 stack, mac80211. The middleware implementation was evaluated experimentally through both functional and performance tests executed on a small multi-hop MANET testbed consisting of nodes equipped with IEEE 802.11 network interfaces. These tests focused on the search functionality provided by the middleware, and the resulting query routing. The results obtained for the various metrics demonstrated that establishing a P2P overlay network on top of a MANET represents a considerable maintenance overhead, even for an unstructured P2P system, when compared to a P2P system which takes advantage of the underlying MANET routing protocol to transport queries and the corresponding results, as is the case of the implemented middleware solution.

References


