Transparent and Distributed Localization of Mobile Users in Wireless Mesh Networks

Mehdi Bezahaf, Luigi Iannone, Marcelo Dias de Amorim, Serge Fdida

Transparent and Distributed Localization of Mobile Users in Wireless Mesh Networks

Mehdi Bezahaf[1], Luigi Iannone[2], Marcelo Dias de Amorim[1], and Serge Fdida[1]

[1] LIP6/CNRS – UPMC Univ Paris 06, France
{bezahaf,amorim,sf}@npa.lip6.fr,
WWW home page: http://www-npa.lip6.fr

[2] Technische Universität Berlin and Deutsche Telekom Laboratories, Germany
luigi@net.t-labs.tu-berlin.de,
WWW home page: http://www.net.t-labs.tu-berlin.de

Abstract. Localization of mobile users in wireless mesh networks (WMN) generally relies on some sort of flooding-based technique. Broadcasting the network is good for reliability but leads to increased latency and broadcast storm problems. This results in low efficiency of the location management mechanism in terms of packets loss and disconnection time. In this paper, we investigate a new DHT-based location management scheme through experimental evaluation on our WMN testbed. The main features of our proposed scheme are that broadcast packets are totally avoided and node localization becomes transparent to the users. We compare it to our previous flooding-based location scheme, namely EMM (Enhanced Mobility Management). Our results show improved performance both in terms of dropped packets and handover latency introduced to re-establish open sessions after a user moves.

1 Introduction

Wireless Mesh Networks (WMNs) are an emerging class of wireless networks that are able to dynamically organize and configure themselves [7]. They allow improving flexibility, efficiency, and coverage, while reducing the complexity of deployment. These characteristics make WMN an attractive solution in many scenarios, including enterprise/home networks, local/metropolitan area networks, and community networks like NYC wireless [5] and Quail Ridge Wireless Mesh Network [19]. Moreover, some industrials have already marketed WMNs [1, 4, 6].

One of the main factors that helped the success of WMNs is its two-tier architecture, inspired from Wireless Local Area Networks (WLANs) and Mobile Ad Hoc Networks (MANET). Indeed, in WMNs there is a clear logical separation between the access subnetwork and the connectivity subnetwork. Wireless Mesh Routers (WMRs) are equipped with at least two different radio interfaces. The first interface is used at the access subnetwork providing connectivity to Wireless Mesh Clients (WMCs) in a WLAN-like fashion. The second interface, configured in ad hoc mode, is used to form a stable wireless backbone providing end-to-end connectivity between clients and with the Internet. In addition

to the architectural inspiration, WMNs have also inherited different communication principles from WLANs and MANETs, including routing protocols and localization services.

Among the various important issues to be tackled, our interest focuses on the localization of mobile WMCs. Localization service consists in maintaining information about the current position of WMCs in the system (i.e., the network topology), and rapidly updating this information with minimum overhead when WMCs move. As previously mentioned, existing localization approaches are mainly inspired from WLAN and MANET networks [8, 11, 22, 23]. As a consequence, the natural two-tier architecture of mesh networks is not exploited and the position of WMCs must be proactively and periodically flooded throughout the network causing scalability issues [25].

In a companion paper, we proposed a localization service based on an on-demand flooding approach, namely EMM (Enhanced Mobility Management) [9]. In EMM, the localization service is separated from the routing protocol and the flooding mechanism is triggered only during a communication setup. A multicast request floods the network each time a lookup is performed. Besides the overhead generated in terms of number of messages, wireless multi-hop networks have proved to be very sensitive to flooding, resulting in very poor performance. To avoid these issues, we need a solution where both lookups and updates are done in a unicast manner.

In this paper, we propose a new distributed localization service for WMNs based on Distributed Hash Tables (DHT). DHTs have shown to be a good alternative in many research domains including domain name services, peer-to-peer file sharing system, decentralized databases, and routing protocols. Surprisingly, we have not seen any DHT-based localization service specifically designed for wireless mesh networks. By taking into account the WMN's two-tier architecture, our DHT-based approach runs only over the backbone, where there is high connectivity and links are relatively stable. In this way, WMCs have no knowledge of the DHT's existence and do not perform any DHT related operation or localization process (*transparency* principle). Each WMR owns a *slice* of the virtual space obtained by hashing the WMRs' identifier. Moreover, each WMR is the *locator* of all WMCs whose identifiers fall within its slice. To this end, we rely on an underlying routing protocol between WMRs, which greatly simplifies the management of the DHT substrate (cf., Section 3).

As a summary, the main features of our approach are:

- **Transparency:** The localization service is totally transparent to the WMCs.
- **Separation:** The localization service takes full advantage the two-tier architecture of WMNs, totally separating the roles of WMCs and WMRs.
- **Overhead Reduction:** The localization service, compared to flooding-based solutions, reduces the overhead by obtaining the positions of WMCs with a single unicast query.
- **Scalability:** The localization service operates correctly even with a large number of mobile WMCs.

– **Robustness:** The localization service avoids flooding messages throughout the network, which results in more robust communications.

The remainder of the paper is organized as follows. In the next section, we describe the motivation of our work, while overviewing EMM, our previous approach. In Section 3, we introduce our DHT-based mechanism, highlighting its main features. Then, in Section 4, we evaluate the performances of the proposed scheme. We conclude the paper in Section 5, summarizing our main results and sketching some future work.

2 Flooding-based location service

While many valuable contributions have been made in order to improve the localization service of mobile users, most of them have been designed in the context of MANETs. Localizing clients in such flat networks is achieved through a routing protocol that relies on the active participation of the clients themselves.

In practice, many existing WMNs reuses solutions originally designed for flat networks, without taking into account the two-tier architecture of WMNs: both WMRs and WMCs share the same localization service, independently of their role in the network. In this case, a node's position (WMR or WMC) is limited to an entry in the routing table (examples of protocols are DSDV [27], OLSR [13], AODV [26], and BATMAN [24]). The performance of these protocols is inherently related to the routing convergence time. Moreover, the complexity of the localization service increases with known routing problems, including the continuous changes in the topology [14], the exposed and hidden terminal problems [10], and broadcast storm problems [25].

In an earlier work, we proposed EMM (Enhanced Mobility Management), a localization service based on an on-demand flooding and where the two-tier architecture of WMNs is taken into account [9]. EMM uses the same principle used in web browser to manage cookies, i.e., WMCs hold information about their latest WMR association in their NDP (Neighbor Discovery Protocol) cache. This information is given to the WMCs by the WMR, in the same way a server provides a cookie to a browser. Thus, when a WMR detects a new WMC association, it extracts from the WMC's NDP cache information about its previous attachment point, and informs the latter about the WMC's movement.

The fact that localization services of both WMRs and WMCs are based on flooding leads to the well-known broadcast storm problem [25]. To avoid the flooding issues and to have a scalable solution, some works propose *rendezvous*-oriented approaches based on a DHT mechanism in ad hoc environments [16]. Each node's unique identifier is related (through a hash function) to one or more other nodes in the system (which maintain the node's current position up-to-date). Inspired from these works, we decide to propose to adapt the same principles to the specific case of WMNs.

As EMM is a reactive localization service, no lookup infrastructure is used to maintain WMCs' current location. All location lookups are achieved by flooding

Table 1. LCTable – Local Clients Table maintains the list of WMCs locally associated to a WMR.

LCTable	
WMC's IP	**WMC's MAC**
Local C_1's IP	C_1's MAC address
Local C_2's IP	C_2's MAC address
⋮	⋮

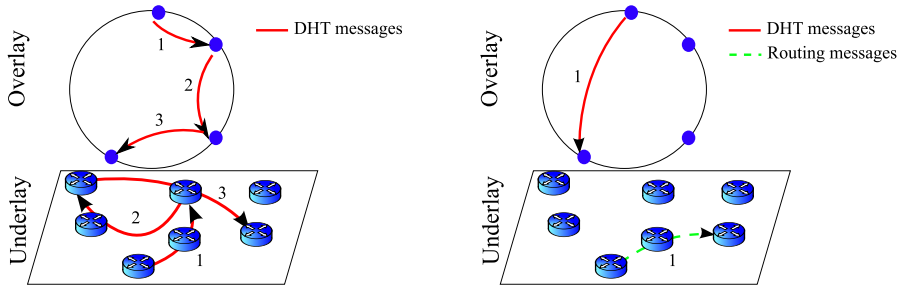
Table 2. FCTable – Foreign Client Table maintains the list of WMCs communicating with its local associated WMCs.

FCTable	
WMCs IP	**IP of WMC's WMR**
Foreign C_1's IP	IP of C_1's WMR
Foreign C_2's IP	IP of C_2's WMR
⋮	⋮

the whole backbone, asking for the destination WMC's location, assuming that the WMR to which the WMC is associated with will reply. Thus, EMM makes the difference between the localization of WMRs, which is still based on a proactive routing protocol (e.g., DSDV) deployed only on WMRs, and the localization of WMCs, which is based on an on-demand flooding approach. To this end, each WMR maintains two tables to manage the communications between its local WMCs and remote WMCs. The first table, called *Local Clients Table (LCTable)*, is used by a WMR to maintain the list of locally associated WMCs (cf., Table 1). This table is automatically filled by the WMR through feedback from the wireless card driver. The second table, called *Foreign Clients Table (FCTable)*, is used by a WMR to maintain the positions of all WMCs (associated with other WMRs) communicating with its local WMCs (cf., Table 2). With such an approach, the respective WMRs of the two communicating WMCs tunnel data packets in the backbone, without the need to inject the position of each client at the WMRs along the path.

A critical operation is the lookup. In EMM, if a WMR wants to reach a specific WMC (not a local one), it sends a multicast request to all other WMRs in the backbone. The current attachment point of the destination WMC replies with a unicast packet. Obviously, the multicast request (which is in fact a broadcast) results in flooding the WMN's backbone.

When a WMC changes its attachment point, EMM uses the NDP (Neighbor Discovery Protocol) protocol [21] to discover the previous attachment point of the mobile WMC [9]. Thereby, the previous attachment point (i.e., the previous WMR) is informed about the WMC's movement. In order to recover open sessions, the new WMR sends for each WMC communicating with the local WMC a multicast message to find their positions. The same is performed by the WMR



(a) Lookup in a peer-to-peer overlay. (b) Lookup in a WMN.

Fig. 1. DHT functionality in two different contexts. Note that in the WMN case, the system relies on an underlying routing protocol running in the backbone.

of the WMCs communicating with the local one, since the old WMR informs them that the WMC moved and a new lookup is necessary.

3 WMC localization as a shared object in a DHT

The fundamental mechanism of a DHT consists in mapping objects' names or identifiers into keys using a hash function, such as SHA-1 [15], and distributing these keys among nodes of the system. Originally motivated by peer-to-peer file sharing systems [2, 3], relying on a DHT as a location substrate also became popular in other network services, such as media streaming (audio, video), instant messaging, domain name services, and decentralized databases, to cite a few [12, 20, 28]). Such a success is mainly due to the attractive DHT properties: totally decentralized architecture, scalability, and robustness.

3.1 Overview of the proposed DHT-based localization service

Contrary to peer-to-peer systems, where DHT is also used to route a request concerning an object toward the server, in our proposal, the connectivity between WMRs is provided by an underlying network routing protocol that is completely independent from the DHT. The latter is only used to manage WMC's localization, mapping WMCs' identifiers into their actual locations. In an example of a file sharing system (cf., figure 1(a)), when a node wants to locate a shared file, it sends a query using the DHT, which is routed in a multi-hop fashion. However, in our WMN, when a node wants to locate a client (cf., figure 1(b)), it infers locally through the DHT the target node (solid arrow in the overlay) and then sends a request to this node through an underlying network routing protocol (dashed arrow in the underlay). This operation is detailed in the following.

The fact that our solution relies on underlying routing protocol in the backbone allows us to use any DHT model; in this work, we decided to use a DHT model based on a virtual ring S inspired from Chord [29]. As the DHT runs only in the backbone, only WMRs manage slices of the addressing space.

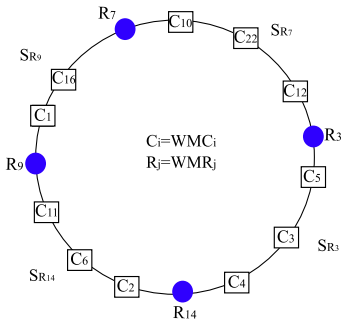


Fig. 2. DHT virtual space organization. It is a ring where the positions occupied by WMRs are indicated by circles and the positions occupied by WMCs are depicted as squares.

Each node (either WMC or WMR) is assigned an identifier obtained by hashing its IPv6 address, as shown in figure 2.[1] Let us call I_{C_i} the identifier of mesh client C_i and I_{R_j} the identifier of mesh router R_j. The hash function we use is SHA-1, which leads to:

$$I_{C_i} = \text{SHA-1}(\text{IPv6}_{C_i}) \ ; \ I_{R_j} = \text{SHA-1}(\text{IPv6}_{R_j}). \tag{1}$$

We define $S_{R_j} \subseteq S$, the space slice containing all WMCs managed by WMR R_j. In other words, R_j is the *locator* of WMC C_i if $I_{C_i} \in S_{R_j}$. Each time a WMC changes its physical attachment point, the new WMR informs the *locator* of this WMC in a unicast fashion. Thus, to obtain the current position of any WMC, a WMR have just to contact the corresponding *locator*.

3.2 Service architecture

In our approach, we keep the same architectural concept of EMM (cf., Section 2). We keep both the LCTable and FCTable but include two additional tables: Managed Clients Table (MCTable), which stores the list of WMCs under the control of the WMR, and Virtual Ring Table (VRTable) for the virtual ring's maintenance.

Virtual Ring Table (VRTable): Stores the list of all WMRs participating in the DHT, in an ascending order. It allows WMRs to determine their successor and predecessor in the ring (see Table 3). Note that this is a straightforward operation. From the underlying routing protocol (which is proactive), every WMR knows all the other WMRs in the network. By applying the hash function on their IP addresses, their position on the ring is easily obtained. Since each node also knows its own position, it also knows the successor and predecessor WMNs on the ring. As a consequence, computing the slice under the control of a node is straightforward. By convention, we consider that the

[1] Our implementation relies on IPv6 instead of IPv4.

Table 3. VRTable – Virtual Ring Table, maintains the information about the partitioned virtual space.

VRTable	
WMR's IP	**Position in Virtual Ring**
IPv6$_{R_1}$	I_{R_1}
IPv6$_{R_2}$	I_{R_2}
⋮	⋮

Table 4. MCTable – Managed Clients Table, maintains the list of WMCs whose identifiers are part of the managed slice.

MCTable		
WMC's IP	**Position in Virtual Ring**	**Location**
IPv6$_{C_1}$	I_{C_1}	IP of WMR with whom C_1 is associated
IPv6$_{C_2}$	I_{C_2}	IP of WMR with whom C_2 is associated
⋮	⋮	⋮

slice managed by a WMR is the share of the ring between its position and the position of its successor.

Managed Clients Table (MCTable): Maintains location information regarding the set of WMCs whose identifiers are part of the slice managed by the WMR (see Table 4). We define MCT$_x$ as the MCTable of mesh router R_x, and MCT$_x$(C_i) as the entry corresponding to client C$_i$ in this table. We will see in the following how this table is filled.

For the sake of robustness, we also implement a redundancy mechanism to deal with WMRs that crash/leave the system. Contrary to P2P systems, where churn is a fundamental problem, backbones in WMN are more stable. For this reason, we assume that redundancy with three replicas is enough. Additionally to its own MCTable, each WMR maintains MCTable of both its successor and predecessor in the virtual ring.

3.3 Protocol Specification

We now present the different operations performed by a WMR w.r.t. the operation of the DHT: join, leave, lookup, and update (when a WMC moves).

WMR joining. When a WMR joins the system, it must fill up its VRTable and obtain a slice of the virtual space. We assume that, at this point, the underlying routing protocol has already performed all the operations on the routing plane (i.e., the WMR is already in the routing table of the other routers and vice-versa). The VRTable is filled with information obtained in the routing table. The node simply hashes all routers' IPs as explained in Section 3.1, obtaining the list of positions occupied in the virtual ring: $[I_{R_1}, I_{R_2}, \ldots, I_{R_M}]$.

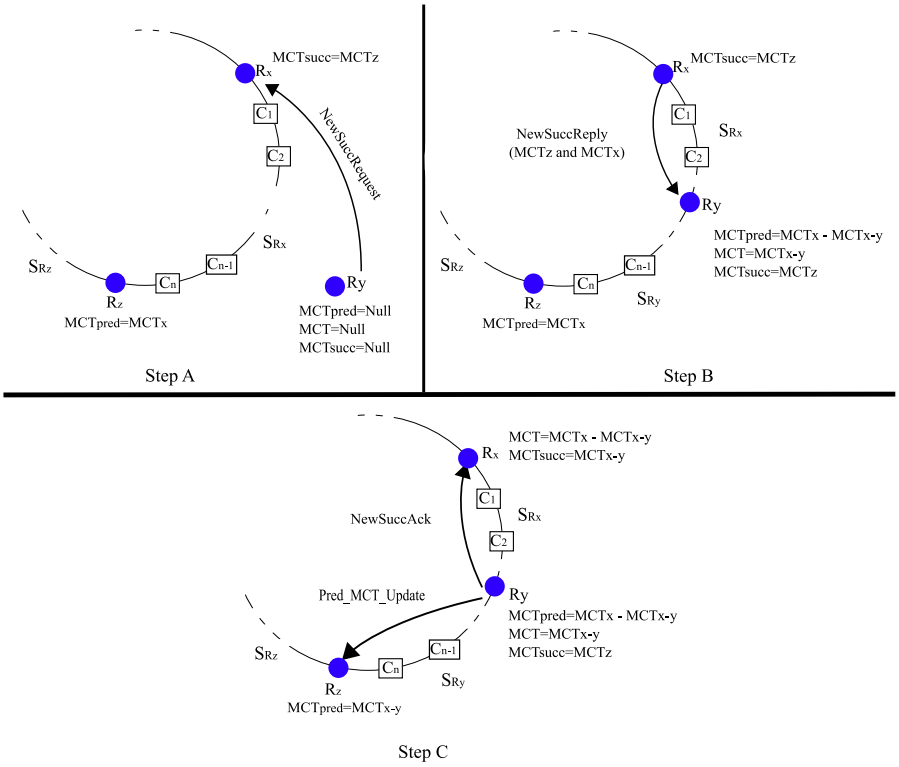


Fig. 3. WMR joining steps

Let R_y be the mesh router joining the DHT. Computing its predecessor and successor is straightforward. They are, respectively, the routers R_x and R_z whose identifiers I_{R_x} and I_{R_z} are immediately inferior and superior to I_{R_y}. This operation is illustrated in figure 3. R_y first informs R_x that it is its new successor in the virtual space (cf., figure 3, step A). R_x replies with both its MCTable (MCT_x) and its current successor's MCTable (MCT_z), which allows the new router R_y to update its MCTable by processing algorithm 1 (cf., figure 3, step B):[2]

Finally, R_y acknowledges R_x and notifies R_z that it is its new predecessor (cf., figure 3, step C). Router R_z replaces its previous MCT_{pred} by MCT_y (since R_y is its new predecessor).

Note that, in order for the above mechanism to properly operate, each time that the routing module detects a change in the network topology the DHT module has to be immediately notified. This allows the corresponding WMR

[2] Although the predecessor responds on behalf of its successor might result is some inconsistency problems, it saves management overhead in the system. We must however guarantee that the redundancy system works fine.

Algorithm 1 Building R_y's MCTables.

```
for C_i ∈ MCT_x do
  if I_{C_i} ≥ I_{R_y} then
    MCT_pred ← MCT_x(C_i)
  else
    MCT_y ← MCT_x(C_i)
  end if
end for
MCT_succ ← MCT_z
```

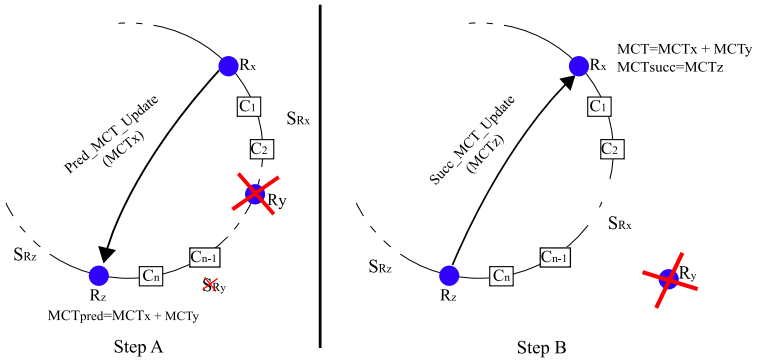


Fig. 4. WMR leaving procedure.

to update the virtual ring. Also note that the notification is necessary for both joining and leaving events.

WMR leaving. When R_x detects that one or more WMRs left (through the underlying routing protocol), it deletes from its VRTable the entry corresponding to the leaving WMR. Then, it checks locally how these changes affect its slice in the DHT virtual ring. If its successor leaves, it must send to its new successor (i.e., its previous successor's successor) its MCTable.

Its new successor locally merges the received MCT_x with its MCT_{pred} (cf., figure 4, step A). It then replies with its own MCTable. Finally, R_x merges its MCTable with its previous successor's MCTable and updates its new successor (cf., figure 4, step B).

Lookup. If a WMR needs to lookup for a specific WMC (not a local one), it starts by hashing the IP address of the WMC in order to find the corresponding *locator*. Then, it sends a unicast request asking the *locator* for the actual position of the WMC. To this end, the *locator* checks its MCTable and replies with current WMC's position (cf., figure 5). Note that this operation only involves unicast messages.

Update. When R_x detects that a new client C_i is now physically connected to its interface (cf., figure 6), it adds C_i's IP address to its LCTable, and notifies

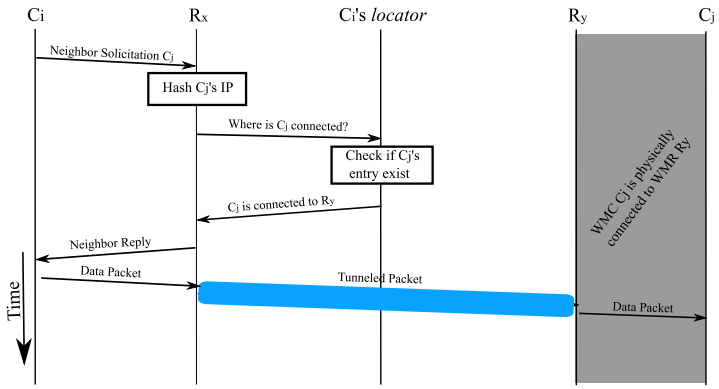


Fig. 5. WMC lookup procedure.

this new association to C_i's locator. The locator, in turn, runs algorithm 2 to check if this client is in its MCTable:

Algorithm 2 Updating C_i's entry at the locator's MCTable.

```
if C_i ∈ MCTable then
    Notify to the previous location of C_i (R_y) that C_i has moved.
    MCTable(C_i).location = R_x's IP
else
    MCTable(C_i) = (C_i's IP, I_{C_i}, R_x's IP)
end if
```

If the entry for C_i is already part of the locator's MCTable, this means that the WMC was already in the network, and it has moved to a different location. Thus, the WMR of the previous location (R_y) must be notified and the information updated. Otherwise, the locator just needs to create a new entry with the new location information. Thus, if C_i moves during communication with a WMC C_j, the WMR of the previous location (R_y) is able to forward correctly traffic to the new location.

4 Evaluation

We now present a number of results obtained through a real implementation of our DHT-based location service.

4.1 Testbed

To evaluate the performance of our approach through real deployment, we implement a Python version of our proposal, and deployed it on MeshDVNet [17].

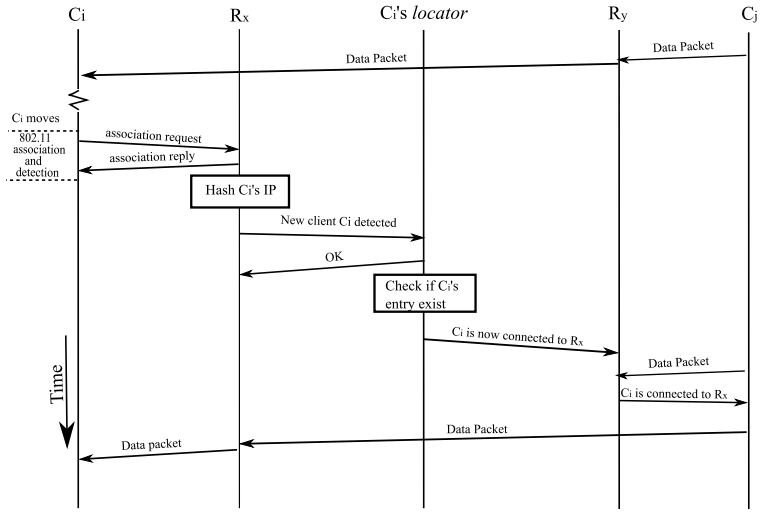
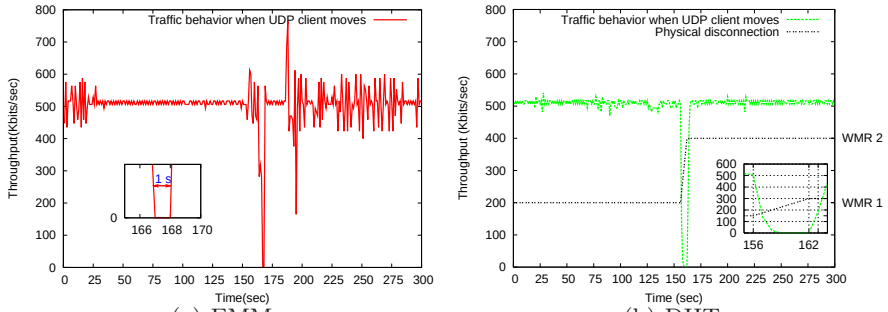


Fig. 6. WMC updating procedure.

Based on IPv6, MeshDVNet offers wireless connectivity to WMCs and allows them communicating as if they were in a traditional wireless LAN (i.e., no changes required at the WMCs). Moreover, to evaluate the efficiency, robustness, and accuracy of our DHT-based solution, we compare our results to EMM, which we recall is a flooding-based approach. We measure the performance of both EMM and our DHT on MeshDVNet using Compaq nx7000 laptops, running Linux Fedora Core 8 (kernel v2.6.24) as WMCs. All WMRs run our DHT v0.1 and Click v1.6.0 over a Linux kernel v2.6.19 on a Soekris board (AMD ElanSC520 133 MHz) with two wireless cards (802.11abg) with Madwifi driver v0.9.4 [18].

4.2 Measurement Setup

We evaluate several different mobility scenarios. We focus on the disconnection time, which represents the duration of traffic disruption that a WMC experiences during handover. To this end, we set up a test configuration where the wireless mobile client C_x changes its physical association from mesh router R_1 to mesh router R_2 while communicating with another client C_y. Three types of traffic were generated during the experiments: (i) unidirectional UDP traffic, (ii) low throughput ICMP (Internet Control Message Protocol) traffic, with its ICMP echo request messages and ICMP echo response messages (generated with the Ping command); and (iii) bidirectional TCP traffic. Concerning the UDP and TCP traffics, they have been generated and measured using the Iperf tool [30].



(a) EMM (b) DHT

Fig. 7. Disconnection time during the handover of a UDP client.

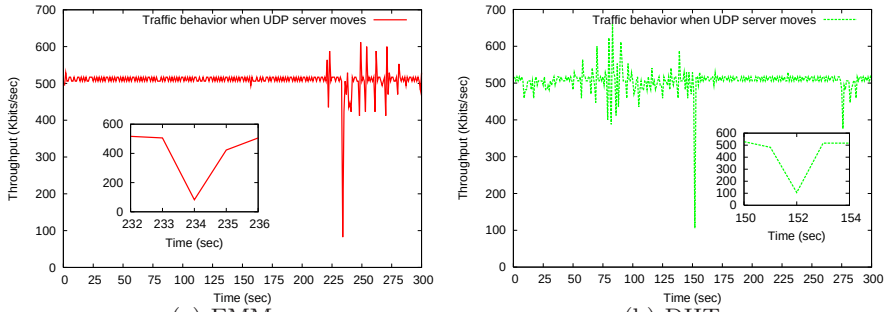
4.3 UDP Unidirectional Traffic

We generated UDP traffic between the C_y (UDP client) and C_x (UDP server). The generated UDP traffic is a unidirectional flow, with short messages without congestion control and without any arriving packet order control. Lost packets are not retransmitted.

Figure 7 shows the disconnection time when C_y moves from R_1 to R_2. In the case of EMM (figure 7(a)), when C_y connects to R_2, R_1 continues to receive C_x's UDP datagrams destined to C_y. As discussed in Section 2, after C_y associates with R_2, the latter notifies R_1 that C_y has changed its location. In turn, this allows R_1, upon the reception of a packet destined to C_y, to notify the sender that the client has moved elsewhere. In our scenario, this means that C_x's attachment point, after receiving this notification, proceeds to a C_y lookup to find its new location by flooding the whole backbone. R_2 replies when it receives the request, updating the location information on the WMR with whom C_x is associated. Then, C_y starts receiving datagrams again.

In our measurements, the IEEE 802.11 handover latency is practically instantaneous (a few milliseconds). We conclude then that when we use a flooding-based approach it takes around one second to recover from traffic disruption after a handover.

In the DHT approach, when C_y connects to R_2, the latter performs a client location update by sending a unicast packet to the C_y's *locator*. At the same time, similarly to the previous case, when C_y associates with R_2, the latter notifies R_1 that C_y has changed its location, allowing R_1 to inform other routers with stale information that C_y has moved. This allows the attachment point of C_x to find C_y's new position by sending a single unicast packet to C_y's *locator*. Figure 7(b) shows the results obtained with the DHT-based approach. The figure should not be interpreted as the worst case; it happens that C_y's card scans all 802.11's channels before connecting to R_2. Such a complete scan lasts 6.34 seconds, while the client is not connected to any router (dashed line). From the figure, it is clear that the throughput increases instantaneously after the physical



(a) EMM (b) DHT

Fig. 8. Disconnection time during the handover of a UDP server.

Table 5. EMM vs. DHT performance when UDP server moves.

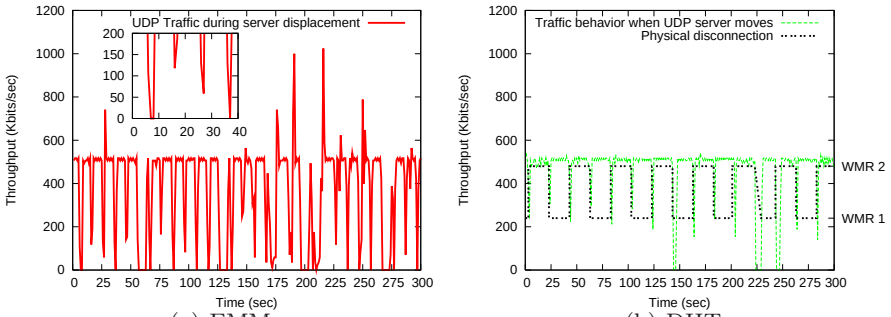
	Duration	Throughput	PDV	% of lost
Flooding	300 secs	491 Kbps	4.417 ms	4%
DHT	300 secs	505 Kbps	3.173 ms	1.3%

connection, meaning that, apart from the long reconnection time of the card, the traffic takes less than one second to recover.

In a second variant of the experiment, we maintain C_y static and move C_x (the server) from R_1 to R_2. In this case, while R_2 notifies R_1, the latter never receives stale traffic for C_x, although C_x is actually sending data. What happens is that when C_x moves its new attachment point performs a client lookup in order to know where to forward the traffic destined to C_y. In the flooding approach, this means that R_2 floods the whole backbone in order to find the new location of C_y. Communication resumes when the attachment point of C_y replies. Figure 8(a) shows that throughput between second 233 and second 234 decreases from 500 Kbits/s to 82.3 Kbits/s. This means that communication is disturbed during less than one second. Between second 234 and second 235, the throughput increases from 82.3 Kbits/s to 420 Kbits/s; during this period no packets are lost.

In the DHT approach, when C_x connects to R_2, the latter performs a WMC location update by sending unicast packet to C_x's locator. Communication resumes when it receives the reply. Figure 8(b) shows that throughput between second 151 and second 152 decreases from 500 Kbits/s to 106 Kbits/s, which means that communication is disturbed over less than one second. Between second 152 and second 153, the throughput increases from 106 Kbits/s to 517 Kbits/s. Again, during this period no packets are lost.

The results presented above are summarized in table 5. We can observe that the DHT approach leads to good performance when compared to the flooding approach. Indeed, not only the average throughput is higher, but also the percentage of lost packet is lower and the packet delay variation (PDV) is lower, leading to reduced and stable disruption times.



(a) EMM (b) DHT

Fig. 9. Disconnection time during repeated UDP server displacements.

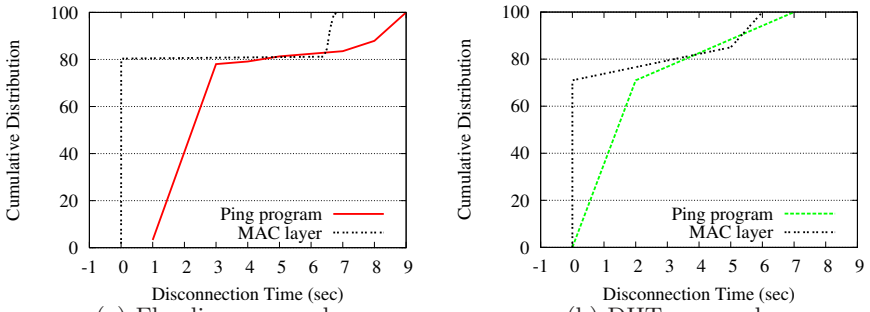
The above results were obtained with a single handover per experiment. We performed as well successive server displacements (one each 20 seconds) in order to have a scenario with increased mobility. The results are shown in figure 9. As it can be seen, traffic disruption time is negligible for the DHT-based approach, lasting less than one second at each handover, whereas for the flooding approach traffic disruption is larger, with severe drops in the throughput. Note that in some cases the disruption time lasts for more than one second due to the driver that chooses to scan all the channels before associating the client with the new WMR.

4.4 ICMP Bidirectional Traffic

In the second set of tests, we evaluated bidirectional communications by sending ping messages between C_x and C_y, with C_x sending ICMP echo requests and C_y replying with ICMP echo responses. Packets are sent at regular interval of one second and that lost packets are not retransmitted.

Figure 10 shows the cumulative distribution function of client disconnection time at both application (Ping) and physical layers (IEEE 802.11 handover). Figure 10(a) shows what happens in the EMM case. For 78.02% of the time, the disconnection period at the physical layer is between 3 and 4 milliseconds, to which corresponds less than 3 seconds of disconnection period at application layer. The maximum disconnection period at the application layer is around 9 seconds, while the maximum disconnection period at physical layer is around 7 seconds. We conclude that using the flooding solution, updating all tables after a handover is time consuming, taking around 2 seconds.

Figure 10(b) shows the results for the DHT case. For 71.43% of the time, the disconnection period in physical layer is less than one second (between 500 and 600 milliseconds), to which corresponds less than 2 seconds of disconnection period at the application layer. Moreover, we can observe that the maximum disconnection period at the application layer is 7 seconds, while the maximum disconnection time at the physical layer is 6 seconds. This leads to the conclusion



(a) Flooding approach (b) DHT approach

Fig. 10. Cumulative distribution function of disconnection time during handover of the ICMP sender.

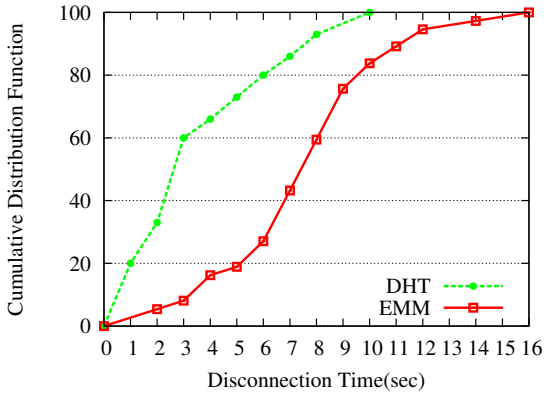


Fig. 11. Cumulative distribution function of Disconnection time during handovers of the TCP sender.

that using the DHT solution, updating all tables after a handover takes around one second. This is 50% less than in the flooding case.

4.5 TCP Bidirectional Traffic

Unlike UDP, TCP guarantees reliability and ordering of packets. This is achieved using the positive acknowledgement mechanism, which consists of retransmitting packets not acknowledged after expiration of a special timer referred as RTO (Retransmission TimeOut). In our tests, client C_x sends TCP packets to C_y. We measure the traffic disruption time in the case of successive handovers (at each 20 seconds). Unlike the tests with UDP, disconnection times during TCP tests are equal or higher than one second. The cumulative distribution function of TCP sender disconnection time during successive handovers, for both EMM and DHT, is shown in figure 11. With DHT, the disconnection time is less than 3

seconds in 60% of the cases, while using EMM it is less than 8 seconds. Moreover, we can observe that maximum disconnection time using our DHT is 10 seconds, while using EMM it is 16 seconds.

5 Conclusion

Wireless mesh networks have gained momentum in the last years and are a candidate as an enabling technology for what is known as the all-wireless Internet. Despite such a success, there are still challenging open issues. In this context, a very important problem is the lack of an effective localization service that guarantees fast and efficient mobility management.

In this paper, we proposed and evaluated through real experiments a DHT solution that relies on a proactive routing protocol running in the backbone. With our approach, we achieve not only the transparency principle but also reduced overhead thanks to the use of unicast messages only. This is orthogonal to existing solutions that rely on flooding-based mechanisms during a lookup procedure. Our results show that, for both unidirectional and bidirectional traffics, the DHT approach leads to significant performance improvements.

As future work, we intend to perform more in-vivo experiments with traffic generated by real applications and a theoretical analysis of the overhead reduction.

References

1. Cisco systems. http://www.cisco.com/.
2. Gnutella. http://rfc.gnutella.sourceforge.net/.
3. Napster, llc. napster. http://www.napster.com/.
4. Nortel. http://www.nortel.com.
5. Nyc wireless. http://www.nycwireless.net.
6. Strixsystems. http://www.strixsystems.com/.
7. I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks Journal (Elsevier)*, 47(4):445 – 487, 2005.
8. Y. Amir, C. Danilov, M. Hilsdale, R. Musăloiu-Elefteri, and N. Rivera. Fast handoff for seamless wireless mesh networks. *ACM Press*, pages 83–95, 2006.
9. M. Bezahaf, L. Iannone, and S. Fdida. Enhanced mobility management in wireless mesh networks. *Journées Doctorales en Informatique et Réseaux (JDIR'08)*, January 2008.
10. V. Bharghavan. Performance evaluation of algorithms for wireless medium access. *In Proceedings of IEEE Performance and Dependability Symposium*, 1998.
11. A. Capone, S. Napoli, and A. Pollastro. Mobimesh: An experimental platform for wireless mesh networks with mobility support. *Proc. of ACM QShine 2006 Workshop on Wireless mesh: moving towards applications*, August 2006.
12. M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik. Scalable distributed stream processing, 2003.
13. T. Clausen and P. Jacquet. *Optimized Link State Routing Protocol (OLSR), RFC 3626*. Internet Engineering Task Force (IETF), October 2003.

14. D. S. J. D. Couto, D. Aguayo, B. A. Chambers, and R. Morris. Performance of multihop wireless networks: shortest path is not enough. *Proceedings of First Workshop on Hot Topics in Networks (HotNets-I)*, October 2002.
15. D. Eastlake and P. Jones. *US Secure Hash Algorithm 1 (SHA1), RFC 3174*. IETF, September 2001.
16. J.-P. Hubaux, J.-Y. Le Boudec, T. Gross, and M. Vetterli. Towards Self-Organizing Mobile Ad-Hoc Networks: the Terminodes Projectæ. *IEEE Comm Mag*, 39(1):118 –124, 2001.
17. L. Iannone and S. Fdida. Meshdv: A distance vector mobility-tolerant routing protocol for wireless mesh networks. *IEEE ICPS Workshop on Multi-hop Ad hoc Networks: from theory to reality (RealMAN'06)*, July 2005.
18. Madwifi Project. Madwifi – multiband atheros driver for wireless fidelity.
19. P. Mohapatra, D. Wu, and D. Gupta. Quail Ridge Wireless Mesh Network: Experiences, Challenges and Findings. *International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, Dec. 2007.
20. M. A. Motoyama and G. Varghese. Crosstalk: scalably interconnecting instant messaging networks. In *WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks*, pages 61–68, New York, NY, USA, 2009. ACM.
21. T. Narten, E. Nordmark, and W. Simpson. *Neighbor Discovery for IP Version 6 (IPv6), RFC 2461*. IETF, Dec. 1998.
22. V. Navda, S. Ganguly, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. Das. Performance optimizations for deploying voip services in mesh networks. *IEEE Journal on Selected Areas in Communication (JSAC)*, Nov. 2006.
23. V. Navda, A. Kashyap, and S. Das. Design and evaluation of imesh: an infrastructure-mode wireless mesh network. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM)*, Jun. 2005.
24. A. Neumann, C. Aichele, M. Linder, and S. Wunderlich. *Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.), draft-openmesh-b-a-t-m-a-n-00.txt Work in Progress*. Internet Engineering Task Force (IETF), March 2008.
25. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, 1999.
26. C. Perkins, E. Belding-Royer, and S. Das. *Ad Hoc On-Demand Distance Vector (AODV) Routing, RFC 3561*. Internet Engineering Task Force (IETF), July 2003.
27. C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *Proceedings of ACM SIGCOMM*, September 1994.
28. V. Ramasubramanian and E. G. Sirer. The design and implementation of a next generation name service for the internet. *SIGCOMM Comput. Commun. Rev.*, 34(4):331–342, 2004.
29. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *In Proceedings of the ACM SIGCOMM 2001 Conference*, August 2001.
30. A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf-the tcp/udp bandwidth measurement tool, 2005.