Enabling Efficient Peer-to-Peer Resource Sharing in Wireless Mesh Networks

Claudia Canali, *Member, IEEE,* M. Elena Renda, Paolo Santi, and Simone Burresi

Abstract—Wireless mesh networks are a promising area for the deployment of new wireless communication and networking technologies. In this paper, we address the problem of enabling effective peer-to-peer resource sharing in this type of networks. Starting from the well-known Chord protocol for resource sharing in wired networks, we propose a specialization that accounts for peculiar features of wireless mesh networks: namely, the availability of a wireless infrastructure, and the 1-hop broadcast nature of wireless communication, which bring to the notions of location-awareness and MAC layer cross-layering. Through extensive packet-level simulations, we investigate the separate effects of location-awareness and MAC layer cross-layering, and of their combination, on the performance of the P2P application. The combined protocol, MESHCHORD, reduces message overhead of as much as 40% with respect to the basic Chord design, while at the same time improving the information retrieval performance. Notably, differently from the basic Chord design, our proposed MESHCHORD specialization displays information retrieval performance resilient to the presence of both CBR and TCP background traffic. Overall, the results of our study suggest that MESHCHORD can be successfully utilized for implementing file/resource sharing applications in wireless mesh networks.

Index Terms—Wireless mesh networks, community networks, distributed hash tables, peer-to-peer resource sharing, cross-layering.

1 INTRODUCTION

WIRELESS mesh networks are a promising technology for providing low-cost Internet access to wide areas (entire cities or rural areas), and to enable the creation of new type of applications and services for clients accessing the network. Differently from other types of wireless multi-hop networks, wireless mesh networks are composed of two types of nodes: mostly stationary wireless access points (*routers*), and mobile wireless clients. Routers are connected to each other through wireless links, and provide a wireless access infrastructure to wireless clients. Some of the routers are connected to the Internet via wired links, and act as gateways for the other routers and for the clients.

Among innovative applications enabled by mesh networking, we mention wireless community networks (see, e.g., the Seattle Wireless initiative [23]), in which users in a community (neighborhood, city, rural area, etc.) spontaneously decide to share their communication facilities (wireless access points) and form a wireless multi-hop network to be used by community members. Wireless community networks can be used to share the cost of broadband Internet access, but also to realize innovative services for the community, such as sharing of community-related resources, live broadcast of local events, distributed backup systems, and so on.

As the above mentioned innovative applications suggest, peer-to-peer resource sharing is expected to play an important role in forthcoming wireless networks based on the mesh technology. In this paper, we investigate the feasibility of the well-known Chord algorithm [24] for peer-to-peer resource sharing in wired networks in a wireless mesh network environment. Starting from the basic Chord design, we propose a specialization – named MESHCHORD – that accounts for peculiar features of mesh networks: namely, i) the availability of a wireless infrastructure, which enables location-aware ID assignment to peers, and ii) the 1-hop broadcast nature of wireless communications, which is exploited through a cross-layering technique that bridges the MAC to the overlay layer.

We evaluate the performance of Chord and MESHCHORD in a wireless mesh network environment through extensive packet-level simulations. The results of the simulations show that MESHCHORD outperforms the basic Chord design both in terms of reduced message overhead for overlay maintenance (mainly achieved by location-awareness), and in terms of increased information retrieval efficiency (mainly achieved by cross-layering). Since communication bandwidth is a limited resource in wireless networks, we evaluate Chord/MESHCHORD performance also in presence of different types of background traffic. We consider congestion-unaware, CBR traffic, and congestion-controlled TCP traffic. To the best of our knowledge, this is the first similar investigation presented in the literature on P2P approaches for wireless networks. The

- *C. Canali is with the Dept. of Information Engineering, University of Modena and Reggio Emilia, Via Vignolese 905, 41100 Modena, Italy. This work was done when Dr. Canali was a Research Assistant at the Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy. E-mail: claudia.canali@unimore.it.*
- *M. E. Renda and P. Santi are with the Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, 56124, Pisa, Italy. E-mail: {elena.renda, paolo.santi}@iit.cnr.it.*
- *S. Burresi is with the Value Team S.P.A., Italy. This work was done when the author was a student of the Dept. of Computer Science, University of Pisa, Italy.*

results of our simulations show that, while Chord information retrieval performance drastically degrades in presence of moderate background traffic, MESHCHORD is relatively resilient to the presence of background traffic, and provides acceptable information retrieval performance also in this situation. Interestingly, the lower message overhead generated by MESHCHORD w.r.t. Chord has a positive effect on congestion-controlled TPC traffic, which observes a relatively higher throughput. Thus, differently from the basic Chord design, MESHCHORD has the potential to provide satisfactory performance in a mixed application environment, where the P2P application coexists with different types of application-layer traffic. Summarizing, the study reported in this paper suggests that MESHCHORD can be successfully utilized for implementing resource sharing applications in wireless mesh networks.

The rest of this paper is organized as follows. In Section 2, we critically discuss related work and this paper's contribution. In Section 3, we present the basic Chord design and our proposed specialization MESHCHORD for wireless mesh network scenarios. In Section 4, we evaluate Chord and MESHCHORD performance through extensive, packet-level simulation. Finally, Section 5 concludes, and outlines possible directions for future work.

2 RELATED WORK AND CONTRIBUTION

Several Distributed Hash Table (DHT) approaches have been proposed in the literature to address the problem of realizing distributed peer-to-peer resource sharing. The various DHT approaches proposed in the literature mainly differ on the structure imposed to the virtual overlay and on the mechanism used to route search requests in the overlay. Among them, we cite Chord [24] (which we briefly describe in the next section), CAN [19], Pastry [21], and Viceroy [14]. However, these DHT approaches have been designed and optimized for operation in wired networks, and issues such as limited bandwidth, node mobility, and so on, are not relevant.

Recent papers have addressed the problem of enabling P2P resource sharing in mobile ad hoc networks (MANETs). Some of them proposed extension/modification of existing P2P approaches to work efficiently on MANETs. Among them, we cite extension/modifications of Gnutella [4], and of Pastry [16], [17]. Others proposed their own solutions, mostly tailored at efficiently dealing with peer mobility. Among them, we cite ORION [12], Mobiscope [6], RBB [25], and the service discovery protocol proposed in [22]. A standard technique used to improve performance of P2P algorithms when used in wireless networks is cross-layering, i.e., taking advantage of information delivered from lower layer protocols (typically, the network layer) when constructing the logical links between peers. The idea is to try to enforce locality as much as possible, i.e., peers which are close in the (logical) overlay topology should be as close as possible also in the physical network topology. Approaches based on this idea are [4], [15], [16]. Although a careful design of the overlay improves the efficiency of P2P systems for MANETs, the combination of node mobility, lack of infrastructure, and unreliable communication medium has hindered the application of P2P approaches in medium to large size ad hoc networks. As a consequence of this, P2P approaches have been successfully applied to MANETs composed of at most a few tens of nodes, and the problem of designing scalable P2P systems for ad hoc networks remains open.

A more recent trend of research pushes the idea of cross-layering a step forward, basically collapsing the overlay and network layer into a unique, location-aware layer, which implements a sort of geographic hash table. This is the case of the approaches proposed in [7], [13], [18]. The technique proposed in [13] is targeted towards MANETs, and is based on the idea of mapping the IDs of the objects to share to trajectories, and to let the nodes which are closer to that trajectory manage the corresponding ID. In [18], the authors propose to use a geographic hash table for in-network storage of archival data in wireless sensor networks. In [7], the authors suggest using a two-tier architecture, where the sensor nodes store the data, and a certain number of proxy nodes implements the distributed indexing mechanism.

Another interesting idea is to integrate the overlay with the network layer, providing a scalable routing scheme with DHT functionality [2], [8]. The motivation is that in very large and dynamic wireless networks, traditional routing mechanisms perform poorly, since they are based either on proactive dissemination of routing information, or on frequent network flooding to discover routes. In the scalable routing concept, nodes are logically organized in a virtual ring (as in a DHT), and proactively maintain routes only to a very limited number of other nodes in the ring (r-successors and r-predecessors in [2], predecessor/successor and a number of Chord-like fingers in [8]). This way, message overhead is considerably reduced with respect to traditional routing mechanisms, and scalability is improved. With scalable routing, messages are routed in a DHT-like fashion: a route request is routed through the virtual ring until a path to the destination is found.

Wireless mesh networks present a potential advantage with respect to MANETs for a successful realization of scalable P2P approaches, namely the presence of a stationary, wireless infrastructure that can be used by mobile clients to communicate with each other (or to access the Internet). Only a few recent papers have explored how P2P approaches can be applied to wireless mesh networks. In [1], the authors evaluate the gain that can be obtained by using network coding when running file sharing applications in wireless mesh networks, and conclude that some gain can actually be achieved, although not as much as in a wired network. In [10], the authors present an incentive model for file sharing in wireless mesh networks. In [9], some of the authors

of this paper introduced a two-tier architecture for file sharing in wireless mesh networks. The main idea is to decouple content user/providers from the distributed resource index: the lower tier is composed of mobile mesh clients, which provide the content (files/resources to share) to the P2P overlay; the upper tier is composed of *stationary* mesh routers, and implements a distributed hash table for locating resources within the network.

The investigation presented in this paper complements our previous work [9] under many respects. While the emphasis in [9] was on the procedures for dealing with client mobility at the lower tier of the architecture, in this paper we are concerned with the efficiency of DHT implementation in the upper tier of the architecture, i.e., stationary mesh routers. Another major difference with respect to [9] is that we consider Chord instead of Viceroy for implementing the DHT, and that we perform *packet-level* simulations to investigate the performance of Chord and of our proposed specialization MESHCHORD in realistic wireless mesh network scenarios. The simulations carried out in the present paper account also for the (considerable) message exchange needed to maintain the Chord overlay, and for dynamic join/leave of nodes at the upper tier of the architecture. On the contrary, the efficacy of GeoRoy (the location-aware version of Viceroy proposed in [9]) was tested only through high level simulations (no MAC layer implementation) performed assuming static topology of the upper tier (no mesh router joining/leaving the network during the simulated time interval), which ignored maintenance overhead. As the results presented in this paper show, *the overhead for overlay maintenance is considerable* and, in some cases, *can lead to network congestion*. Hence, *results obtained from high level simulations that ignore maintenance overhead and do not implement the MAC layer can be very inaccurate* (this applies not only to the results presented in [9], but also to most of simulation-based results presented in the literature). Finally, differently from [9], we investigate the performance of information retrieval in terms of percentage of successful queries and query response time, also in presence of background traffic.

Another major contribution of this paper is the notion of cross-layering that we exploit in the MESHCHORD design: while existing works exploit cross-layering to extract information from the network layer (typically, IDs of physical neighbors of a peer node) to improve performance [4], [15], [16][1], in MESHCHORD *we extract information from the MAC layer.* The main idea is to exploit the "wireless advantage" (1-hop broadcast nature of wireless communications) to possibly capture packets which are not destined to a certain peer node u, but for which u possesses relevant information (e.g., u stores the key requested for in the packet). This technique proves very useful in improving the information retrieval performance and in increasing the number of successful join operations[2] in highly dynamic networks, while only marginally increasing (if not decreasing) the total number of packets circulating in the network. To the best of our knowledge, MESHCHORD is the first proposal exploiting MAC cross-layering for improving performance in P2P file sharing applications for wireless networks.

Finally we want to cite [5], where the authors present a (packet-level) simulation-based investigation of Chord performance in MANETs environments. The authors consider three different routing protocols, and show that Chord performance is only marginally influenced by the choice of the routing protocol. The main finding of [5] is that node mobility considerably impairs Chord consistency, with a dramatic effect on information retrieval performance: in presence of even moderate mobility, the percentage of successful queries can drop below 10%.

There are several differences between our study and the one reported in [5]. The application scenario is different: stationary mesh networks instead of MANETs. Most importantly, in our work we consider not only the basic Chord design, but also a location-aware, cross-layer specialization of Chord for stationary mesh networks. Our analysis shows that these improvements to the original design are indeed fundamental to provide satisfactory performance in wireless mesh networks, especially when background traffic is present. Hence, the main finding of this work is that our proposed specialization MESHCHORD can successfully be applied in a wireless mesh network, also in scenarios in which the P2P application coexists with different types of application-layer traffic.

3 MESHCHORD

In this section, we shortly describe the two-tier architecture used in our design, and the basic Chord operations. We then describe in details MESHCHORD, our proposed specialization of Chord for wireless mesh networks.

3.1 Network architecture

Similarly to [9], we assume a two-tier architecture for file/resource sharing (see Figure 1): the lower tier of the architecture is composed of (possibly) mobile mesh clients (clients for short), which provide (and use) the content to be shared in the P2P system; the upper tier of the architecture is composed of stationary mesh routers, which implement a DHT used to locate file/resources within the network. Unless otherwise stated, in the following we use the term *peer* to refer exclusively to a mesh router forming the DHT at the upper tier of the architecture.

We assume routers are stationary and plugged, but they can be switched on/off during network lifetime. This is to account for situations that might arise in some mesh network application scenarios (e.g., community

1. Note that we also exploit this information in MESHCHORD, but we use the term 'location-awareness' instead of cross-layering to refer to this technique.

2. As explained in the following, join operations are not necessarily successful.

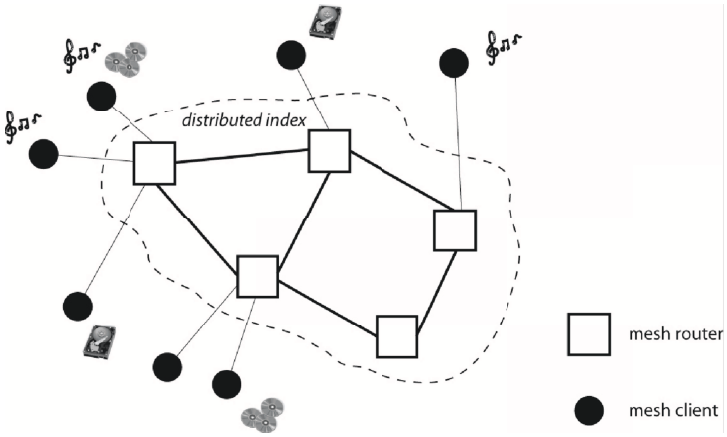

Fig. 1. Two-tier architecture assumed in our design.

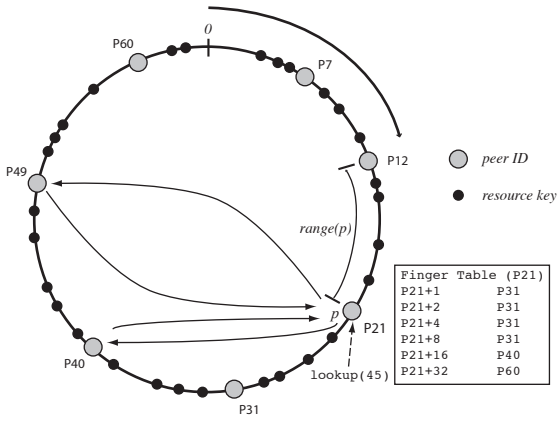

Fig. 2. Basic Chord operations. m is set to 6.

networking), in which routers might be managed by users and occasionally shut down. Also, changes in the upper tier topology might be caused by failures of some router.

Mesh clients are the content users and providers: they share file/local resources with other mesh clients, as well as access resources shared by others. When a client u wants to find a certain resource, it sends to its responsible router a (a mesh router within its transmission range) a *FindKey* message, containing the key (unique ID) of the resource to find (see next section for details on key assignment to node/resources). The responsible router forwards the resource request in the DHT overlay according to the rules specified by the Chord protocol (see below), until the resource query can be answered. In case of successful query resolution, a message containing the IP address of the client holding the requested file/resource is returned to client u through its responsible router a. For details on the rules for responsible router selection, and on the procedures needed to deal with client mobility (handoff between responsible mesh routers, locating a mobile client in the network, etc.), and to add/remove shared resources to/from the distributed index, the reader is referred to [9].

3.2 Basic Chord operations

The DHT approach investigated in this paper is Chord [24]. Chord is based on the idea of mapping both peer (mesh router) IDs and resource IDs (keys) into the same ID space, namely the unit ring $[0, 1]$. Each key resides on the peer with the smallest ID larger than the key (see Figure 2), i.e., peer p manages keys comprised between its own ID and the ID of the predecessor of p in the unit ring (denoted $range(p)$). Associated with each key is the IP address of the mesh client holding the corresponding resource. Chord maps peer and resource IDs into the unit ring using a hashing function, named $Sha1$, which has the property of uniformly distributing IDs in $[0, 1]$. Indeed, IDs in Chord are represented through m-bit numbers, i.e., at most 2^m distinct (peer or resource) IDs are present in the Chord system. In the following, we set $m = 24$, which corresponds to having about 16 millions possible IDs. This is a reasonable ID space for wireless mesh networks, in which the number of shared resources is expected to be in the order of several thousands, and the number of mesh routers (peers) in the order of a few hundreds. Indeed, larger ID spaces can be easily included in our design with no modification, and with negligible impact on performance. This allows dealing with larger networks and number of shared resources, which might lead to a non-negligible probability of conflicting ID assignment with relatively low values of m (see also Section 4).

The main operation implemented by Chord is the $lookup(x)$ operation, which can be invoked at any peer to find the IP address of the peer with ID$= x$ if x is a peer ID, or the IP address of the peer responsible of key x in case x is a resource ID. $lookup$ operations are used both for query resolution ($FindKey$ operation) and for overlay maintenance.

To speed up $lookup$ operations, every peer maintains a table of up to m distinct peers (fingers). The i-th finger of peer j, with $1 \leq i \leq m$, is the peer which has the smaller ID larger than $j + 2^{i-1}$. Note that some of the fingers (especially for low values of i) can actually coincide (see Figure 2). In order to facilitate join/leave operations, each peer maintains also the ID of its predecessor in the Chord ring (peer $P12$ for peer $P21$ in Figure 2).

When a $lookup(k)$ operation is invoked at peer p and the operation cannot be resolved locally (because k is not within $range(p)$), a message is sent to the peer p' with largest ID $< k$ in the finger table of node p. If p' cannot resolve the $lookup$ operation, it replies to peer p with a message containing the ID of the peer p'' with largest ID $< k$ in its own finger table. Peer p then forwards the request to peer p'', and so on, until the $lookup$ operation

can be resolved (in at most m steps)[3]. Referring to Figure 2, a lookup operation for key 45 issued at node P21 is first forwarded to node P40, and then to node P49, which is responsible for the key and can resolve the $lookup$.

To deal with dynamic join/leaves of peers in the systems, the following procedures are implemented. When a new peer p joins the network, it first needs to initialize its predecessor and finger table. This is done by sending requests to any peer currently joining the network peer p is aware of (called *hook* peer). Then, the finger tables and predecessor pointers of currently active peers must be updated to account for the new peer joining the network. Finally, peer p must contact its successor s in the ring so that the key range previously managed by s can be split with p. In case no (active) hook peer can be found, the join operation fails, and the peer cannot join the Chord overlay. When an existing peer p leaves the network, it first informs its predecessor and successor in the ring about its intention of leaving the network, so that they can change their finger tables and predecessor pointers accordingly; then, peer p transfers to its successor the key range it is responsible for.

Finally, we mention that, in order to deal with dynamic network conditions, each active peer in the network periodically performs a $Stabilize$ operation, which verifies and possibly updates the content of the finger table and predecessor pointer. The period between consecutive $Stabilize$ operations is a critical parameter in the Chord design: if the period is relatively short, the network is more reactive, but a higher message overhead is generated; on the other hand, a longer stabilize period reduces message overhead, at the expense of having a less reactive network. How to set the stabilize period in order to satisfactorily address this tradeoff when Chord is executed in wireless mesh networks is carefully investigated in Section 4.

3.3 Location-awareness

The first modification we propose to the basic Chord design concerns the function used to assign ID to peers (hash function $Sha1$ is still used to assign key to files/resources). The idea is to exploit locality, and to assign peers which are close in the physical network with close-by IDs in the unit ring. This choice is motivated by the observation that, according to Chord specifications, most of the messages are exchanged between a peer and its successor/predecessor in the unit ring.

More specifically, location-awareness is implemented by assigning IDs to peers according to the following function (see [9]):

$$ID(x,y) = \begin{cases} \frac{x\Delta}{s^2} + \lfloor \frac{y}{\Delta} \rfloor \cdot \frac{\Delta}{s} & \text{if} \lfloor \frac{y}{\Delta} \rfloor \text{ is even} \\ \frac{(s-x)\Delta}{s^2} + \lfloor \frac{y}{\Delta} \rfloor \cdot \frac{\Delta}{s} & \text{if} \lfloor \frac{y}{\Delta} \rfloor \text{ is odd} \end{cases},$$

3. This corresponds to the iterative method for implementing $lookup$ operations in Chord [24]. Also recursive $lookup$ implementation is considered in the original Chord design. A comparison between iterative and recursive $lookup$ implementation in MESHCHORD is reported in [3].

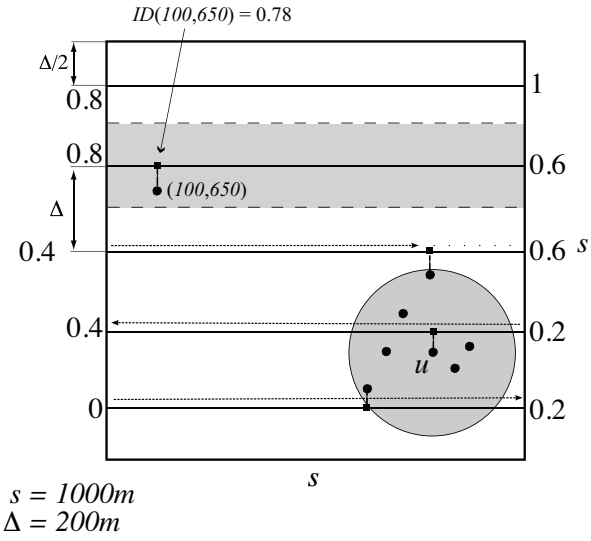

Fig. 3. Location-aware mapping of peer IDs and interaction with cross-layering.

where $ID(x,y)$ is the ID of a peer with coordinates $(x,y) \in [0,s]^2$, s is the side of the deployment region, and Δ is a parameter which defines the 'granularity' of location-awareness: the lower the value of Δ, the closer the peers must be in the physical network in order to be mapped into close-by regions of the unit ring. Figure 3 shows an example of location-aware ID assignment when $s = 1000m$ and $\Delta = 200m$. The deployment region is divided into $s/\Delta = 5$ sub-regions of width Δ and length s (shaded area). A segment of the unit ring bisects each sub-region, with alternate left/right orientation. This is required to ensure that peers in the same physical vicinity and close to the border of the deployment region are mapped to close-by segments of the unit ring (see Figure 3). Each bisecting segment of the unit ring has virtual length Δ/s. Peers in the same sub-region of the deployment area (shaded area) are mapped to the same segment of the unit ring, thus converting physical proximity into proximity in the unit ring. In the example reported in Figure 3, a peer located at coordinates $(100, 650)$ is assigned ID 0.78 in the unit ring.

Note that the above location-aware ID assignment function requires that peers are aware of their location, which can be easily accomplished in wireless mesh networks through, e.g., the use of GPS receivers.

3.4 Cross-layering

The second contribution of the MESHCHORD proposal concerns the introduction of a MAC cross-layering technique. This technique aims at speeding up the $lookup$ operations by exploiting the information that is available at the MAC layer due to the 1-hop broadcast communication occurring in wireless networks. The basic idea is that a peer u may capture packets for which it owns relevant information, even if they are not destined to

u. This technique is motivated by the possibility that peer u, that may actually be able to resolve a $lookup$ request, is physically close to the peer invoking the $lookup$ operation, while they are far away in the unit ring.

More specifically, whenever a peer u receives a packet at the MAC layer, u sends it up to the application layer for further processing, even if the packet was not destined to u. If the packet does not contain a $lookup$ request, it is discarded. Otherwise, u checks if it may resolve the $lookup(x)$ operation. This occurs if x is comprised between u's ID and the ID of the predecessor of u in the unit ring. In this case, u sends a message containing its own ID to the peer that invoked the $lookup(x)$ operation. It is important to note that, since the $lookup$ process is invoked for both query resolution and overlay maintenance, cross-layering may improve the performance of both these operations.

3.5 Interactions between location-awareness and cross-layering

Location-awareness is designed to map neighboring peers to close-by IDs in the unit ring. On the other hand, cross-layering tends to be more effective when physical neighbor peers have far-away IDs in the unit ring (this results in a higher likelihood of "capturing" packets). Hence, an ideal ID mapping function should, for a certain peer u, both $i)$ map $pred(u)$'s and $succ(u)$'s IDs to peers which are physical neighbors of u, and $ii)$ assign the remaining u's physical neighbors far-away IDs in the unit ring. The design of a mapping function which achieves both $i)$ and $ii)$ is a very complex combinatorial problem, which is left for future work. In this paper, we focus on property $i)$, and show that, even in presence of location-aware peer ID assignment, cross-layering is indeed beneficial to the performance of the P2P application. This can be explained referring back to Figure 3. In the lower right corner of the deployment area, the transmission range of a peer u is shaded. It is easy to see that, if the transmission range and parameter Δ are adequately set (in the figure, $\Delta = TxRange = 200m$), although most u's physical neighbors are actually assigned close-by IDs in the unit ring, we do have a number of u's physical neighbors whose IDs are far away in the unit ring. These physical neighbors have opportunities for capturing packets generated by/destined to peer u, explaining the relative benefits of cross-layering that can be observed also in presence of location-aware ID assignment (see figures 7 and 10).

3.6 MESHCHORD analysis

Using techniques similar to those reported in [9], we provide a theoretical characterization of MESHCHORD performance in random networks, which are formally defined as follows:

Definition 1. *In a random network deployment, n peers are distributed uniformly at random in a square region* $R = [0, s)^2$. *The peers have transmission range* r, *with* $r = 2s\sqrt{\frac{2 \log n}{n}}$.

The specified value of r in the above definition guarantees that the resulting network is connected w.h.p.[4] (see [9]). We first show that MESHCHORD location-aware ID assignment preserves the property of uniformly distributing IDs in the unit ring. This result implies that the nice properties of the original Chord design (load balancing on the overlay, query resolution in $O(\log n)$ steps, etc.) are preserved in MESHCHORD.

Lemma 1. *Assume a random network deployment; then, the peer IDs computed according to mapping* $ID(x, y)$ *are distributed uniformly at random in the [0,1] interval (unit ring).*

Proof: The proof is along the same lines as proof of Theorem 1 in [9]. □

We now turn to analyzing the *stretch factor*, which is formally defined as follows:

Definition 2. *Given a lookup operation on key* k *on an overlay network O, let* $l(k)$ *be the hop distance in the physical network between the peer at which the lookup is invoked and the peer that manages the key range to which* k *belongs; furthermore, let* $P(k)$ *be the path traversed by the lookup*(k) *in the overlay network, and let* $l(P(k))$ *be the hop length of* $P(k)$ *in the physical network. The stretch factor is defined as:*

$$stretch(O) = \max_k \left\{ \frac{l(P(k))}{l(k)} \right\} . \quad (1)$$

Informally speaking, the stretch factor measures how close a virtual overlay is to the topology of the underlying network. The next theorem provides an upper bound to MESHCHORD stretch factor in random networks:

Theorem 1. *Assume a random network deployment; then,* MESHCHORD *stretch factor is* $O(\sqrt{n \log n})$ *w.h.p.*

Proof: Given Lemma 1 and Chord's properties, we have that a $lookup$ operation is resolved traversing $O(\log n)$ hops in the virtual overlay, w.h.p. By Lemma 1 of [9], under the assumption of random network deployment, each hop in the virtual overlay corresponds to $O\left(\sqrt{\frac{n}{\log n}}\right)$ hops in the physical network, w.h.p. Hence, a $lookup$ operation is resolved traversing $O(\sqrt{n \log n})$ hops in the physical network, w.h.p. We now observe that, given a key k, $l(k) \geq 1$ whenever the peer issuing the $lookup(k)$ operation cannot resolve the query locally. It is easy to see that, given a random key k, Lemma 1 implies that the probability of locally resolving a $lookup(k)$ converges to 0 as $n \to \infty$. It follows that $l(k) \geq 1$ w.h.p., and the theorem is proved. □

Note that the bound stated in Theorem 1 is the same holding for the GeoRoy protocol of [9]. This means that, in asymptotic terms, we can expect comparable performance in terms of message overhead between

4. W.h.p. means with probability approaching 1 as $n \to \infty$.

the two protocols. However, MESHCHORD implements also cross-layering, which is not considered in GeoRoy. Hence, we expect that MESHCHORD performs considerably better than GeoRoy in terms of information retrieval performance and resiliency to inconsistencies in the finger tables (see Section 4).

4 PERFORMANCE EVALUATION

We have evaluated the performance of Chord and of the proposed specialization MESHCHORD on mesh networks using GTNetS, a packet-level network simulator developed at Georgia Institute of Technology [20]. To better understand the contribution of location-awareness and MAC cross-layering on Chord performance, we have also considered a version of Chord in which only location-awareness is implemented (ChordLoc), and a version of Chord in which only MAC cross-layering is implemented (ChordXL).

The starting point of our work was a pre-existing implementation of Chord developed by the Network Security and Architecture Lab at Georgia Tech, which, however, did not implement join/leave procedures and, more importantly, it had never been tested on wireless networks. Hence, in a first phase we had to fix several bugs in order to have a perfectly working implementation of Chord in wireless mesh networks.

We considered two network topologies in simulations:

- *grid*: peers are located in a square, equally spaced grid; peer separation is $100m$;
- *random uniform*: n peers are distributed uniformly at random in a square area of side s, where $s = \sqrt{n} \cdot 100m$.

In both cases, we assume peers are equipped with $802.11b$ radios, the link data rate is $11Mbs$, and radio signal obeys free space propagation. For routing messages between far-away peers, we used the DSR routing algorithm [11]. This choice is motivated by the fact that energy consumption, which is not optimal for DSR, is not an issue in our target application scenario. Most importantly, DSR is one of the most popular routing algorithm for wireless multi-hop networks, and that the simulation-based analysis of [5] has shown that DSR performs best among the various routing protocols considered when used in conjunction with Chord (although in a MANET scenario). Note that scalable source routing algorithms (see [2], [8]) can be used in combination with our P2P algorithms. However, given the specific application scenario we consider (stationary networks of moderate size with a modest peer churn rate), we do not expect significant benefits from using scalable source routing instead of DSR. In fact, as discussed in Section 2, scalable source routing performance benefit over traditional routing mechanisms becomes considerable in very large, dynamic networks. For instance, the extensive simulation results reported in [2] show that VRR (the approach proposed in [2]) displays performance similar to DSR in our reference scenario of stationary networks of moderate size.

To model dynamic join/leaves of peers into the network (which occur only after all peers have initially joined the network, and the Chord overlay is stabilized), we assume that each peer updates its status every $30sec$, possibly making a transition to active/inactive state. More specifically, a peer which is currently active becomes inactive with probability p_{leave}, while a currently inactive peer becomes active with probability p_{join}. By acting on the values of p_{leave} and p_{join}, we can carefully tune the number of join/leave events in the network. Unless otherwise stated, in the following we assume $p_{leave} = p_{join}$.

When a peer u leaves the network, it notifies its successor u' by sending it key range $range(u)$. Considering that in our proposed architecture the information associated to a key is the IP address of the mesh client storing the requested resource, that IP addresses are very short (4 bytes), and that key IDs are 24 bits long, each entry in the key table is composed of 56 bits. Given this, we have implemented key range transfer trough the communication of a single UDP message with a $2KB$ payload between the leaving peer and its successor, which is sufficient to transfer as many as 292 entries in the key table. Note that, even if we use longer IDs to deal with a larger number of peers and shared resources, we can still pack a large number of key table entries in a single UDP message. For instance, with $m = 48$, we can pack as many as 204 key table entries is a single UDP message.

When a peer u (re-)joins the network, it first has to find an active peer u' it is aware of (*hook peer*). The selection of the hook peer is different depending on whether the peer joins the network for the first time. If a peer u has already been part of the network in the past, it stores the IP address of the last three fingers in its finger table (this is because the last fingers in the table are most likely distinct – recall Figure 2) before leaving the network. Peer u then tries to contact the first finger in the list, then, in case it is not responding, the second one, and so on, until u is able to join the network, or the join operation fails. In case of the first join, the finger table is empty, and the above procedure cannot be performed. Hence, upon the first join each peer uses the IP address of a special peer (the first node joining the network, which is always the same) as hook peer. In the bootstrapping phase, peers join the network one at the time at fixed time intervals ($1sec$) using the special hook peer, thus allowing setting up and stabilizing the finger tables. Overall, this bootstrapping phase takes $200sec$ in our simulations.

If the join operation is successful, and after the successor pointer is stabilized, peer u sends a message to its successor u'', so that u'' can send to u a part of its key range. This is also implemented through communication of a single UDP message with $2KB$ payload.

A certain number of queries is generated during

Chord lifetime. Queries are generated uniformly over time (every t_{query} seconds); when a new query is generated, the peer that issues the query is chosen uniformly at random among the currently active peers, and the ID of the key k to be searched is chosen uniformly at random in [0,1] (expressed as an m-bits binary number).

In the following, we present the results of four different sets of simulations, focusing on the effect of $i)$ increasing the number n of peers, $ii)$ changing the number of join/leave events in the simulated time interval, $iii)$ changing the query rate, and $iv)$ different types of background traffic, on MESHCHORD performance. For all the sets of experiments, the simulated time interval was $3800sec$, where the first $200sec$ were used to incrementally add peers to Chord, and to stabilize the overlay; all the simulation results presented in the following refer to data gathered in the last $3600sec$ of the simulation, and are averaged over 10 (50) runs in case of grid (uniform random) topology. The largest sample set in case of random topologies was needed to account for the higher degree of randomness (which also plays a role in determining the network topology) in this setting. The simulation parameters are summarized in Table 1.

Topology	grid/random
numb. peers	$49 \div 144$
depl. region side	$1000m$
Radio technology	802.11b
Link data rate	$11Mbps$
TxRange	$200m$
Δ	$200m$
$1 - p_{leave}$ $(= 1 - p_{join})$	$0.9 \div 0.999$
query rate	$2 \div 60$ queries/min/peer
sim duration	$3600(+200)secs$
background traffic	CBR/TCP
CBR rate	$20Kbs \div 200Kbs$

TABLE 1
Simulation parameters.

The performance of the various versions of Chord considered in our simulations is expressed in terms of:

- *message overhead*: total number of network-level packets exchanged by Chord to maintain the overlay, and to resolve the queries;
- *query resolution*: percentage of queries which are successfully resolved; a query on key k is successfully resolved if the IP address of the peer responsible for key k is returned to the peer which issued the query;
- *query response time*: for successful queries, the time elapsed between the instant the query is issued by peer p, and the instant the answer is received at peer p;
- *successful join ratio*: percentage of successful join over all join operations; the percentage is computed accounting only for the join operations occurring after the initial stabilization period.

4.1 Preliminary simulations

In a preliminary set of simulations, whose results are not shown, we have optimized a set of parameters such as node transmission range, value of Δ in the location-aware versions of Chord, and the stabilize interval.

We have verified that, in case of grid topology, setting the transmission range to $200m$ (i.e., twice the node separation) is the best choice for reducing Chord overhead. For uniformity, we have used the same transmission range also in the case of random uniform topology. Note that in this case disconnected network topologies might arise; since Chord operation is not guaranteed in case the underlying network is disconnected, we have discarded these topologies in our simulations.

We have also verified that the values of Δ resulting in the better performance (in terms of message overhead) are in the order of the transmission range. For this reason, we have set $\Delta = 200m$ in the following experiments.

Finally, we have investigated the optimal setting for the stabilize interval which, we recall, is the interval between consecutive invocations of the $Stabilize$ operation by a peer. This parameter has a major effect on Chord performance, since it determines the control overhead generated to maintain the overlay. We have experimentally verified that if the stabilize interval is too short (in the order of $1 - 3secs$), the network is close to congestion (using the basic version of Chord) when only Chord-related messages are exchanged. On the contrary, if the stabilize interval is too long ($10secs$ or higher), the Chord overlay is not responsive enough, and several queries and join operations fail. We have verified through extensive simulations that setting the stabilize interval to $7.5secs$ results in the best compromise between message overhead and network responsiveness.

4.2 Increasing network size

In the first set of experiments, we considered networks of size ranging from 49 to 144 peers (mesh routers). This is an appropriate range of sizes for our target application scenario, in which the DHT overlay is realized over the stationary infrastructure nodes of a wireless mesh network. The number of queries during the simulated time interval is fixed at $120 \cdot n$, corresponding to having (on the average) each node generating a new query every $30sec$. We consider both a relatively static network configuration, in which the average up/down time of a peer is $8hr$, and a very dynamic network configuration, in which the average up/down time of a peer is $5min$. We remark that peers in our scenario are mesh routers part of the wireless infrastructure, hence the above churn rates are deemed adequate to represent static and dynamic network conditions.

Figure 4 reports the total number of network-layer packets exchanged during the simulated time interval in the grid topology. Location-awareness is very effective in reducing message overhead: the reduction can be as high as 40% in the almost static scenario, while it is

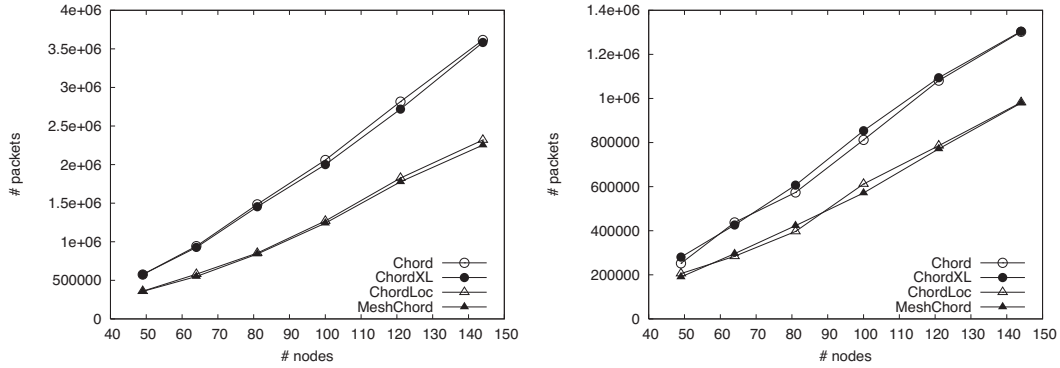

Fig. 4. Total number of packets exchanged in grid networks for increasing number of n: almost static network (left) – $(1 - p_{leave}) = 0.999$, and very dynamic network (right) – $(1 - p_{leave}) = 0.9$.

somewhat lower in the very dynamic scenario. On the other hand, cross-layering seems to have only marginal effect on message overhead, if not actually increasing the number of exchanged messages in very dynamic networks. Indeed, this higher overhead is caused by a very positive effect of cross-layering, which is depicted in Figure 5. As seen from the plots, cross-layering significantly increases the number of successful join operations, especially in case of very dynamic networks: while the number of successful join operation is consistently above 95% with cross-layering, with the original Chord protocol this percentage can drop to as low as 70% in case of very dynamic networks. We believe this is due to the fact that cross-layering mitigates the negative effects of having inconsistent finger tables, which tend to increase the percentage of unsuccessful join operations. Inconsistencies in the finger tables are clearly more likely to occur under dynamic network conditions, which explains the relatively greater benefits of cross-layering on the percentage of successful join operations under such conditions. Hence, the higher message exchange observed with cross-layering in dynamic networks is caused by the larger number of peers joining the Chord overlay in this situation.

It is also worth observing that location awareness tends to *decrease* the number of successful join operations under very dynamic network conditions. We believe this is due to the fact that, while location-awareness is very effective in reducing message overhead (Figure 4), its effect on *lookup* operations (which are at the basis of both new join and query resolution procedures) can actually be detrimental. In fact, as explained in Section 3.5, location-aware ID assignment tends to rule out the possibility of having close-by peers in the physical network which are far-away in the unit ring. This negative effect of location-awareness becomes more evident for larger networks (see Figure 5-right).

The results for the random topology scenario confirm the trends observed in the grid scenario, with a somewhat lower reduction in terms of message overhead of MESHCHORD with respect to Chord (as high as 30%). This lower reduction is due to the fact that cross-layering is even more effective in increasing the percentage of successful join operations in case of random network topology than with grid topologies.

Figure 6 reports the percentage of unsuccessful queries in very dynamic networks ($(1 - p_{leave} = 0.9)$). MESHCHORD tends to slightly decrease the query success rate with respect to the original Chord; however, this apparent shortcoming of MESHCHORD is indeed caused by the fact that MESHCHORD tends to operate on a larger network (recall Figure 5), where the success rate is physiologically smaller. The results obtained in the almost static scenario showed an above 99% query success rate even for the basic Chord protocol.

Figure 7 reports the average query response time in very dynamic networks. As seen from the plots, cross-layering has a significant effect on response time in both grid and random topologies. On the other hand, location-awareness only achieves marginal reductions with respect to the basic Chord protocol. Overall, MESHCHORD achieves as high as 60% reduction in query response time with respect to Chord. The results for the almost static scenario show very similar trends.

4.3 Varying the number of join/leaves

In the second set of experiments, we fixed the size of the network to $n = 100$, and considered different values of p_{leave}. The results reported in Figure 8 show that location-awareness considerably reduce message overhead, over the entire range of p_{leave} values considered. These results are in accordance with those reported in Figure 4. It is also worth observing that the relative advantage of MESHCHORD over Chord tends to become smaller for more dynamic networks; as observed in the previous section, this is due to the increased number of successful join operations achieved by cross-layering.

Figure 9 clearly shows the decreasing trend of the percentage of unsuccessful queries as the network becomes less and less dynamic: while the success rate is above 96% for almost stationary networks, it becomes as low as 93% for more dynamic networks. This trend is less pronounced when the network topology is random.

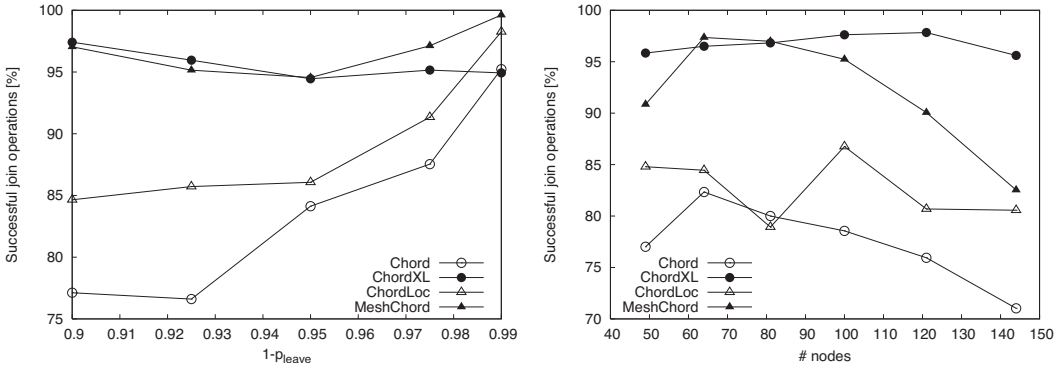

Fig. 5. Percentage of successful join operations in grid networks for $n = 100$ and increasing values of $(1 - p_{leave})$ (left), and for increasing value of n with $(1 - p_{leave}) = 0.9$ (right).

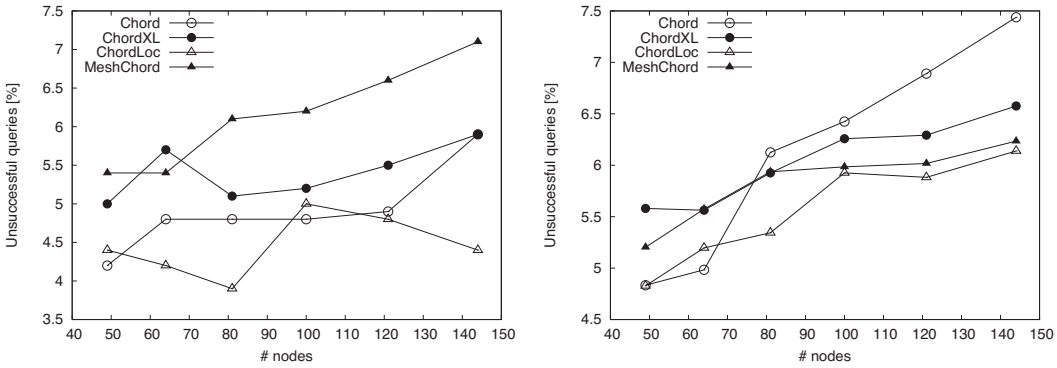

Fig. 6. Percentage of unsuccessful queries in very dynamic networks for increasing values of n: grid topology (left), and random topology (right).

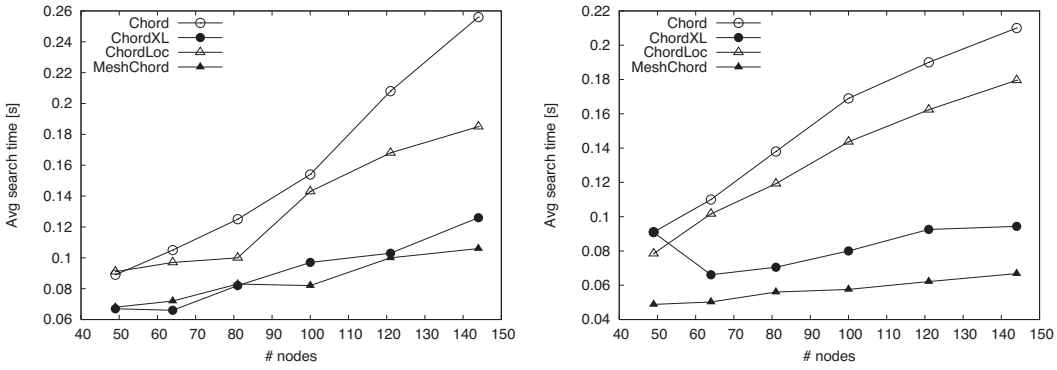

Fig. 7. Average query response time in very dynamic networks for increasing values of n: grid topology (left), and random topology (right).

Finally, Figure 10 shows that MESHCHORD, due to cross-layering, is very effective in reducing the query response time with respect to Chord over the entire range of p_{leave} values considered.

4.4 Varying the number of queries

In the third set of experiments, we have fixed $n = 100$ and set $(1 - p_{leave}) = 0.9$, and investigated the relative performance of the various versions of Chord considered with an increasing number of queries (up to 60 query/min per node on the average). The average query response time of the various versions of Chord with increasing query rate is reported in Figure 11.

As seen from the figure, MESHCHORD outperforms the basic Chord design under all investigated range of query rates. The relative advantage of MESHCHORD with respect to Chord becomes larger with increased query rate. In particular, in the case of grid topology the average query response time is reduced from $0.2secs$ to less than $0.1secs$ (50% reduction) with the lowest query rate, while the reduction is from $1.78secs$ to $0.65secs$ (62% reduction) with the highest query rate. A similar

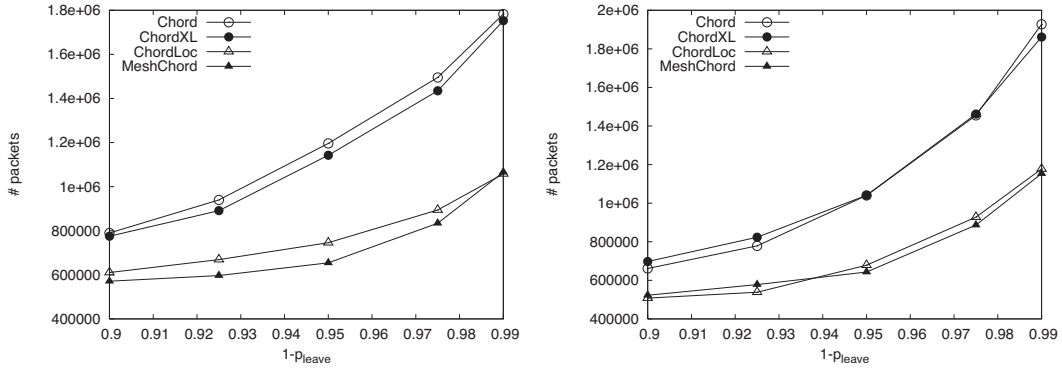

Fig. 8. Total number of exchanged packets for different values of $(1 - p_{leave})$: grid topology (left), and random topology (right).

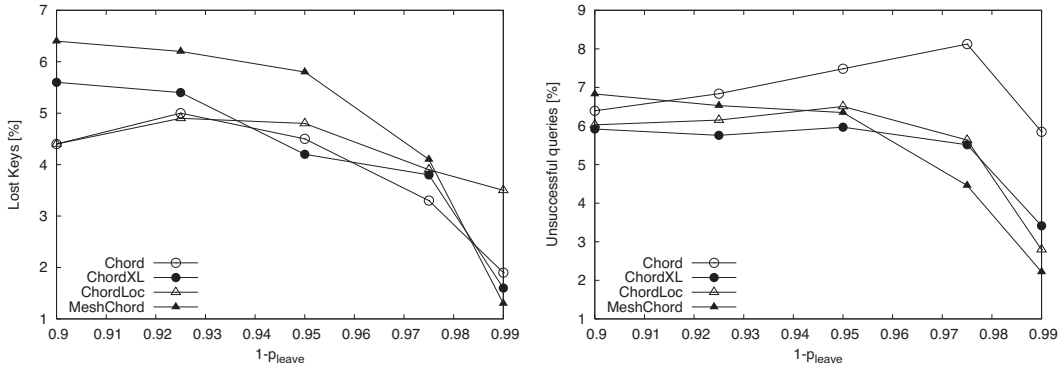

Fig. 9. Percentage of unsuccessful queries for different values of $(1 - p_{leave})$: grid topology (left), and random topology (right).

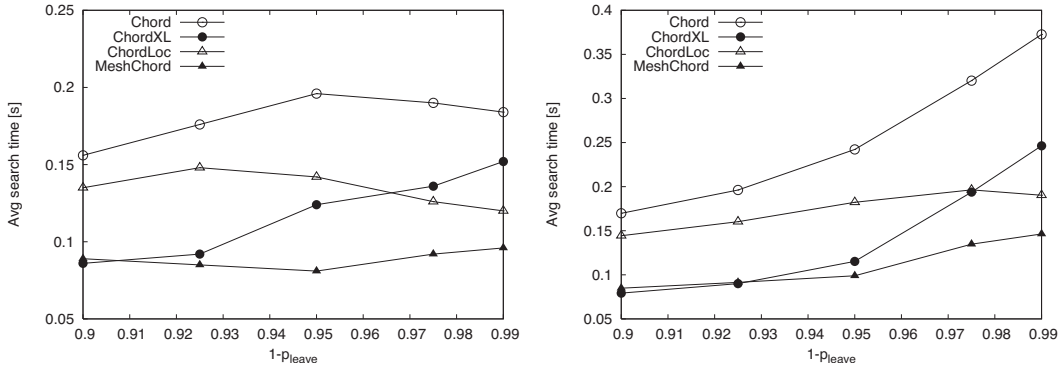

Fig. 10. Average query response time for different values of $(1 - p_{leave})$: grid topology (left), and random topology (right).

trend is observed with the random topology. Consistently with the previous results, slightly higher search times are observed in the random topology with respect to the grid topology. It is worth observing that the average query response time with MESHCHORD remains below $1sec$ even with the highest query rate. Finally, the results of this set of simulations have confirmed that the reduction in the average query response time is mainly due to MAC cross-layering, since ChordXL performance is relatively close to that of MESHCHORD, especially for relatively low query rates.

4.5 Background traffic

In the fourth set of experiments, we have evaluated the performance of the basic Chord and of our proposed MESHCHORD specialization in presence of different types of background traffic. More specifically, we have considered two types of background traffic, one oblivious to the congestion level in the network (CBR) and the other congestion-aware (TCP). We have then evaluated how Chord and MESHCHORD performance scales with the number n of peer nodes. In this specific set of simulations, we have restricted our attention to

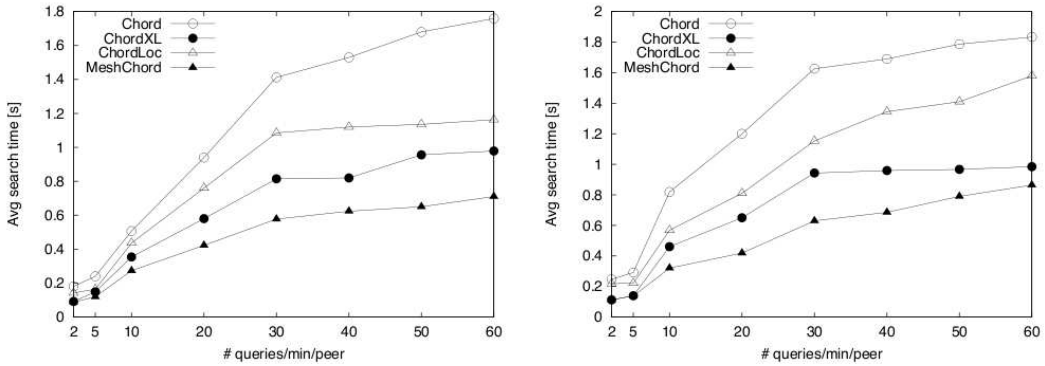

Fig. 11. Average query response time for increasing query rate: grid topology (left), and random topology (right).

grid topologies, and initially assumed static networks ($p_{join} = p_{leave} = 0$)

Let us first consider the case of CBR (Constant Bit Rate) traffic. For a given network with n nodes, we have randomly selected $\sqrt{n}$ CBR flows with rate $80Kbs$. The resulting Chord and MESHCHORD performance is reported in Figure 12–left (failed key searches) and –right (query response time). For the sake of comparison, we have also included the curves when no background traffic is present in the network. As seen from the figure, contrary to the basic Chord design, MESHCHORD is able to preserve acceptable performance in presence of background traffic even for relatively large networks: when $n = 144$, less than 15% of the searches fail with MESHCHORD, while more than 35% of the search operations fail with Chord. Also the average query response time when using MESHCHORD is considerably reduced with respect to the basic Chord: from about $2.5sec$ to less than $1.5sec$ when $n = 144$. It is also worth observing the considerable impact of CBR background traffic on Chord and MESHCHORD performance: when $n = 144$, the percentage of failed search operations increases from about 1% to about 35% with Chord, and from nearly 0 to about 15% with MESHCHORD; the average query response time increases from about $0.4sec$ to about $2.5sec$ with Chord, and from about $0.15sec$ to about $1.4sec$ with MESHCHORD.

In Figure 13, we have considered a specific network size ($n = 100$), and varied the rate of the CBR flows from $20Kbs$ to $200Kbs$. As seen from the figure, the performance of MESHCHORD is always superior to the one of Chord, especially for rates ranging from $20Kbs$ to $160Kbs$; for higher rates, the network is congested, and MESHCHORD can provide only marginal improvements with respect to Chord for what concerns the average query response time, while it still provides substantial improvements in terms of percentage of failed search operations.

Let us now consider the case of TCP traffic. Similarly to the previous scenario, in a network with n peer nodes we have randomly selected $\sqrt{n}$ random source/destination pairs, and established TCP flows between them. The main difference between CBR and TCP traffic lies in the congestion control mechanism implemented in TCP: i.e., the sending rate is automatically adjusted (with an additive increase, multiplicative decrease law) to match the current network load, which is estimated through explicit ACK packets sent back from the destination to the source node. The effect of congestion control can be clearly seen from the plots reported in Figure 14, which reports the percentage of failed key searches and the average query response time as a function of n: differently from the case of CBR traffic, Chord/MESHCHORD performance do not consistently decrease as n increases; instead, performance worsen until n is around 100, and it stabilizes, if not even improves, for larger values of n. This effect is due to the TCP congestion control mechanism: as the network is relatively small, the relatively few TCP flows observe a lightly loaded network, and can increase the sending rates up to the maximal values. However, as the network becomes larger and the number of active TCP flows (as well as Chord/MESHCHORD-related traffic) increases as well, a certain level of network congestion is observed, and the sending rates of the TCP flows are significantly reduced. Thus, the actual amount of background traffic is stabilized (if not reduced), and Chord/MESHCHORD performance stabilizes as well (if not improves). The above described effect of the TCP congestion control mechanisms can be clearly deduced from Figure 15, which reports the average number of KB sent for each active TCP flow as n increases. As seen from the figure, as the network size increases, the average number of KB sent per flow tends to decrease, owing to the additive increase, multiplicative decrease rule used to adjust sending rates. When $n = 144$, the active flows are almost starved.

It is also worth observing that MESHCHORD consistently and considerably outperforms Chord also in presence of TCP background traffic: in the most critical network conditions ($n = 100 - 121$), MESHCHORD percentage of failed search operations is around 13%, with an average query response time of less than $1.2sec$, which should be compared with 27% and $1.8sec$, respectively, provided by Chord. Another notable feature of MESHCHORD is that its lower message overhead (with

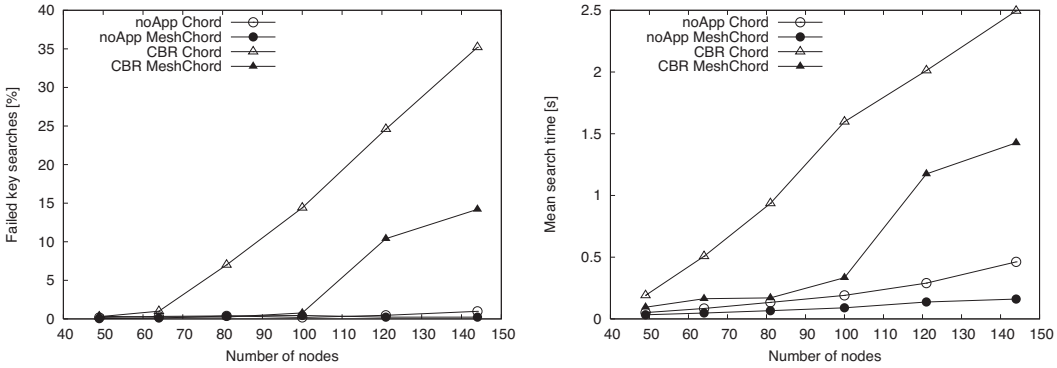

Fig. 12. Percentage of failed key searches (left) and average query response time (right) in presence of CBR background traffic.

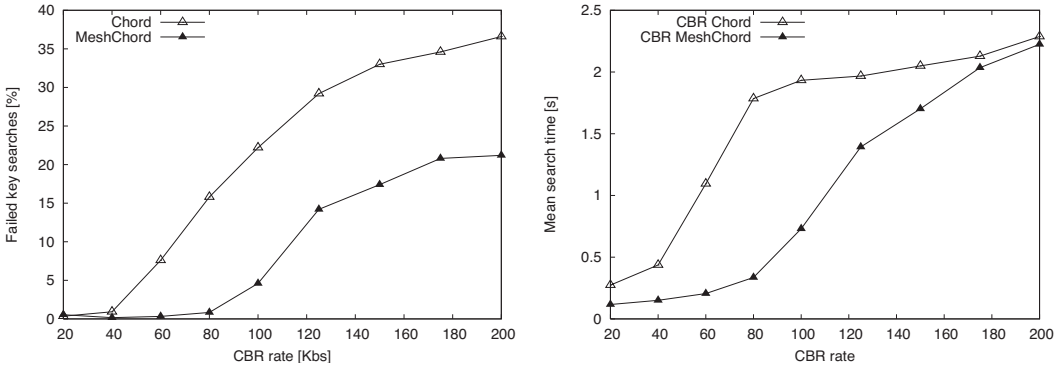

Fig. 13. Percentage of failed key searches (left) and average query response time (right) for increasing CBR rates, with $n = 100$ peer nodes.

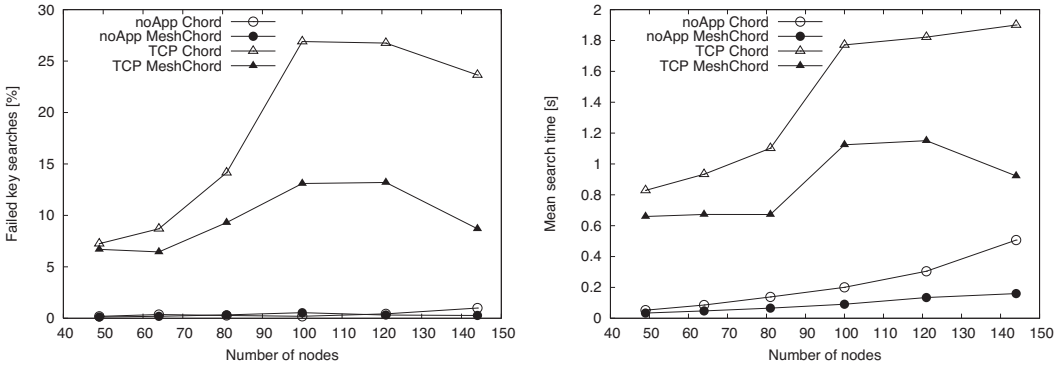

Fig. 14. Percentage of failed key searches (left) and average query response time (right) in presence of TCP background traffic.

respect to Chord) has a positive effect not only on the performance of the P2P application itself, but also on the performance of the applications generating background traffic: from Figure 15, we can see that MESHCHORD increases the average throughput of TCP flows of as much as 30% with respect to the basic Chord design.

We have repeated the above experiments in presence of dynamic network conditions, i.e., when peer nodes join/leave the network at different times. In particular, we have considered a setting in which $(1 - p_{join}) = (1 - p_{leave}) = 0.9$, corresponding to highly dynamic network conditions. Note that, due to the presence of background traffic, we impose that peers which are also source/destination of background traffic flows remain active for the entire simulated time interval.

Chord and MESHCHORD performance in presence of CBR background traffic is reported in Figure 16. As seen from the figure, MESHCHORD considerably reduces the percentage of failed search operations with respect to Chord also in this case, although to a somewhat lower extent. In terms of average query response time, though, MESHCHORD performs worse than Chord for the largest

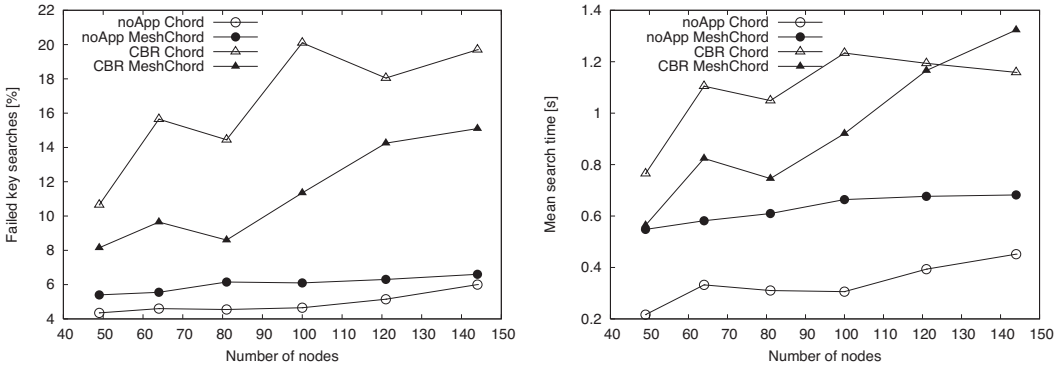

Fig. 16. Percentage of failed key searches (left) and average query response time (right) in presence of CBR background traffic under dynamic network conditions.

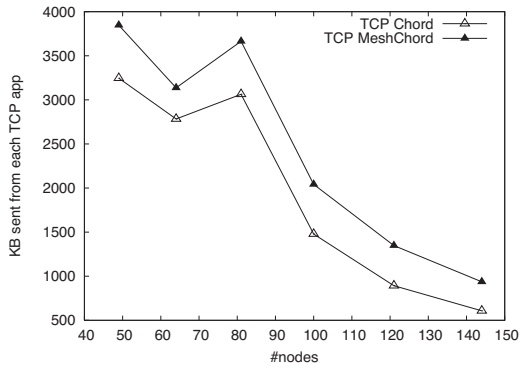

Fig. 15. Average number of KB sent per TCP flow for increasing network size.

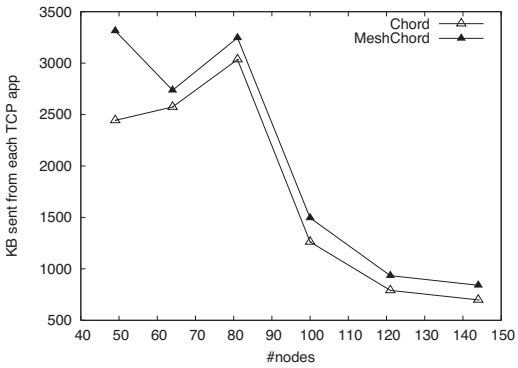

Fig. 18. Average number of KB sent per TCP flow for increasing network size under dynamic network conditions.

network size ($n = 144$). This is due to the fact that, under dynamic network conditions, the actual overlay network size with MESHCHORD is considerably larger than with Chord, due to the higher number of successful join operations achieved by MESHCHORD (recall Figure 5). Hence, queries tend to traverse a larger number of hops with MESHCHORD, which results in a relatively higher average query response time. Again, we stress that this apparent performance degradation provided by MESHCHORD in presence of large, dynamic networks, must indeed be interpreted as an evidence that MESHCHORD is more effective than Chord in maintaining a large and coherent P2P overlay.

Figure 17 reports Chord and MeshChord performance in presence of TCP background traffic. Differently from the previous case, MESHCHORD consistently outperforms Chord in terms of both percentage of failed searches and average query response time. It is also worth observing the much greater impact that background traffic has on Chord performance, with respect to the similar impact on MESHCHORD performance: for instance, when $n = 100$, the percentage of failed searches with Chord increases from about 4.5% in case of no background traffic to about 19% with TCP traffic, while MESHCHORD displays a much lower increases (from 6% to 10%); for what concerns average query response time, it increases from $0.3sec$ to $1.3sec$ with Chord, and only from $0.65sec$ to $0.8sec$ with MESHCHORD. This clearly indicates that, contrary to Chord, MESHCHORD performance is relatively resilient to the presence of background traffic (note that a similar trend is displayed also in presence of CBR background traffic – see Figure 16).

Finally, Figure 18 reports the average number of KB sent for each TCP flow in presence of dynamic network conditions. Similarly to the case of static networks, the effect of the TCP congestion control mechanism can be clearly deduced, as well as the better performance of TCP applications achieved by MESHCHORD with respect to the basic Chord design.

4.6 Discussion

Summarizing, the results of our study have shown that, differently from what happens in a MANET scenario [5], Chord can be effectively used for implementing a distributed file/resource index in wireless mesh networks, at least when the co-existing traffic is low. Most importantly, Chord performance can be considerably improved by accounting for specific features of the considered application scenario, namely precise knowledge

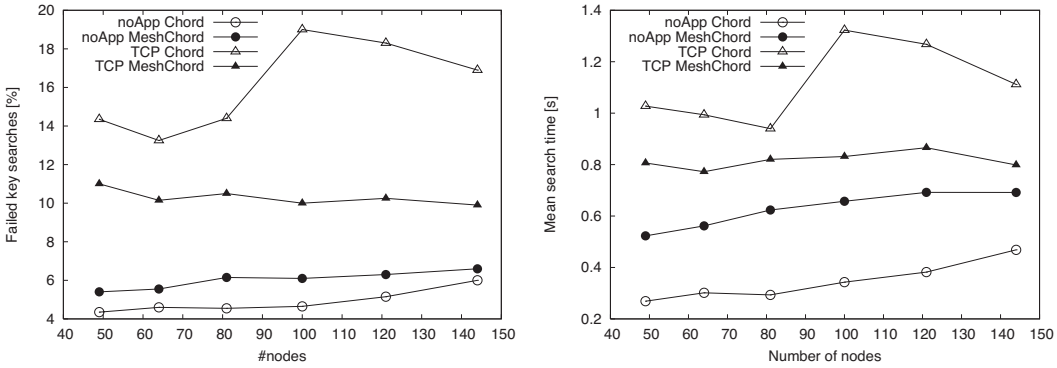

Fig. 17. Percentage of failed key searches (left) and average query response time (right) in presence of TCP background traffic under dynamic network conditions.

of peer location (location-awareness) and 1-hop broadcast nature of communications (MAC cross-layering). Location-awareness proves very useful in reducing message overhead, of as much as 40% with respect to the basic Chord desing. Cross-layering, on the other hand, is very effective in mitigating the negative effects of inconsistencies in the finger tables, leading to a considerable increase in the number of peers successfully joining the network. Cross-layering has also a beneficial effect on the query response time, which is significantly reduced with respect to the basic Chord design. The only price to pay is a marginal increase (less than 2%) in the percentage of unsuccessful queries. The protocol resulting from the combination of these two techniques, MESHCHORD, sums up the relative advantages provided by location-awareness and cross-layering over Chord, and considerably outperforms the basic Chord design under all respects.

MESHCHORD performance gain with respect to Chord is much more evident in presence of background traffic: contrary to Chord, MESHCHORD is able to provide satisfactory performance of the P2P overlay also in presence of background traffic, while in turn increasing the performance of the applications contributing to the background traffic (e.g., increasing the average throughput of TCP flows). In general, MESHCHORD performance appears to be relatively resilient to the presence of background traffic (as long as the network is not congested), which is not the case for Chord.

5 CONCLUSIONS

In this paper, we have carefully investigated through packet-level simulation the performance of the Chord approach for peer-to-peer resource sharing in wireless mesh networks. We have also proposed a specialization of the basic Chord approach called MESHCHORD, which exploits peculiar features of wireless mesh networks (location-awareness, and 1-hop broadcast nature of wireless communications) to improve performance.

The main finding of the study reported in this paper is that, contrary to what happens in MANET environments [5], the Chord approach can be successfully utilized for implementing file/resource sharing applications in wireless mesh networks. However, the basic Chord design is effective only under relatively static network conditions and in presence of modest background traffic. With respect to the basic Chord design, our proposed MESHCHORD protocol achieves a considerable reduction in message overhead, and a significant improvement in information retrieval performance. This performance improvement allows an effective realization of the P2P overlay also under very dynamic network conditions and in presence of considerable background traffic.

Although our investigation has shown that MESHCHORD message overhead does not lead to network congestion by itself, overlay maintenance still requires the exchange of a relatively high number of messages in the network, which could induce performance degradation when other applications are executed concurrently with MESHCHORD. Quantifying application-layer performance degradation when several applications coexist with the P2P overlay is matter of ongoing work, as well as the problem of further reducing the message overhead induced by applications for file/resource sharing in wireless mesh networks.

REFERENCES

[1] A. Al Hamra, C. Barakat, T. Turletti, "Network Coding for Wireless Mesh Networks: A Case Study", *Proc. IEEE Int. Symposium on a World of Wireless, Mobile and Multimedia (WoWMoM)*, 2006.
[2] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, A. Rowstron, "Virtual Ring Routing: Network Routing Inspired by DHTs", *Proc. ACM SIGCOMM*, pp. 351–362, 2006.
[3] C. Canali, M.E. Renda, P. Santi, "Evaluating Load Balancing in Peer-to-Peer Resource Sharing Algorithms for Wireless Mesh Networks", *Proc. IEEE MeshTech*, pp. 603–609, 2008.
[4] M. Conti, E. Gregori, G. Turi, "A Cross-Layer Optimization of Gnutella for Mobile Ad Hoc Networks", *Proc. ACM MobiHoc*, May 2005.
[5] C. Cramer, T. Fuhrmann, "Performance Evaluation of Chord in Mobile Ad Hoc Networks", *Proc. ACM MobiShare*, pp. 48–53, 2006.
[6] M. Denny, M. Franklin, P. Castro, A. Purakayastha, "Mobiscope: A Scalable Spatial Discovery Service for Mobile Network Resources", *Proc. International Conference on Mobile Data Management (MDM)*, 2003.
[7] P. Desnoyers, D. Ganesan, P. Shenoy, "TSAR: A Two Tier Sensor Storage Architecture Using Interval Skip Graphs", *Proc. ACM SenSys*, Nov. 2005.

[8] T. Fuhrmann, "Scalable Routing for Networked Sensors and Actuators", *Proc. IEEE SECON*, pp. 240–251, 2005.
[9] L. Galluccio, G. Morabito, S. Palazzo, M. Pellegrini, M.E. Renda, P. Santi, "Georoy: A Location-Aware Enhancement to Viceroy Peer-to-Peer Algorithm", *Computer Networks*, Vol. 51, n. 8, pp. 379–398, June 2007.
[10] L. Huiqiong, D. Xuyang, L. Hansheng, W. Wenmin, "P2P File Sharing in Wireless Mesh Networks", *Proc. Conference on Advanced Parallel Processing and Techniques*, LNCS 4847, pp. 402-413, 2007.
[11] D.B. Johnson, D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", *Mobile Computing*, n. 353, pp. 153–181, 1996.
[12] A. Klemm, C. Lindemann, O.P. Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks", *Proc. IEEE VTC-Fall*, Oct. 2003.
[13] O. Landsiedel, S. Gotz, K. Wehrle, "Towards Scalable Mobility in Distributed Hash Tables", *Proc. IEEE Conf. on Peer-to-Peer Computing*, 2006.
[14] D. Malkhi, M. Naor, D. Ratajczak, "Viceroy: A Scalable and Dynamic Emulation of the Butterfly", *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*, Jul. 2002.
[15] G. Moro, G. Monti, "W-Grid: a Cross-Layer Infrastructure for Multi-Dimensional Indexing, Querying and Routing in Wireless Ad Hoc and Sensor Networks", *Proc. IEEE Conf. on Peer-to-Peer Computing*, 2006.
[16] A. Passarella, F. Delmastro, M. Conti, "XScribe: a Stateless, Cross-Layer Approach to P2P Multicast in Multi-Hop Ad Hoc Networks", *Proc. ACM MobiShare*, pp. 6–11, 2006.
[17] H. Pucha, S.M. Das, Y.C. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks", *Proc. IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2004.
[18] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, F. Yu, "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table", *Mobile Networks and Applications*, Vol. 8, No. 4 pp. 427–442, 2003.
[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network", *Proc. ACM Sigcomm*, Aug. 2001.
[20] G. Riley, "The Georgia Tech Network Simulator," *ACM SIGCOMM MoMeTools Workshop*, 2003.
[21] A. Rowstron, P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large Scale Peer-to-Peer System", *Proc. IFIP/ACM Middleware (LNCS 2218)*, pp. 329–350, 2001.
[22] F. Sailhan, V. Issarny, "Scalable Service Discovery for MANET", *Proc. IEEE PerCom*, 2005.
[23] `http://www.seattlewireless.net/`
[24] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc. ACM Sigcomm*, Aug. 2001.
[25] O.Wolfson, B. Xu, H. Yin, H. Cao, "Search-and-Discover in Mobile P2P Network Databases", *Proc. IEEE ICDCS*, 2006.

Claudia Canali is assistant professor at the University of Modena and Reggio Emilia since 2008. She got the Laurea Degree summa cum laude in computer engineering from the same university in 2002, and the Ph.D. Degree in computer engineering from the University of Parma in March 2006. During the Ph.D. she spent eight months as visiting researcher at the AT&T Research Labs in Florham Park, New Jersey. Her research interests include distributed architectures for Internet-based services, content delivery networks and mobile systems for mobile Web access. On these topics, she published about twenty articles on international journals and conferences. She is member of IEEE Computer Society. For additional information: http://weblab.ing.unimo.it/people/canali.

M. Elena Renda received the Laura Degree in Computer Science with full marks from the University of Pisa, Italy, in 2000, and the Ph.D. Degree in Information Engineering from the Scuola Superiore Sant'Anna di Studi Universitari e di Perfezionamento, Pisa, Italy, in 2009. She has been a research associate at the Istituto di Scienza e Tecnologie dell'Informazione - CNR in Pisa, from 2001 till 2005, and, since 2005, a research associate at the Istituto di Informatica e Telematica - CNR in Pisa. During the Ph.D. she spent seven months as visiting researcher at the Language Technologies Institute of Carnegie Mellon University, Pittsburgh, PA. Her research interests during Ph.D. included several aspects of information retrieval, such as personalization, information filtering, meta-search, collection and document fusion, automatic source selection, and schema matching. More recently, her interests have broadened including P2P resource sharing protocols for wireless mesh community networks, and efficient and scalable algorithms for DNA pattern repetitions identification and extraction. She has published several articles on international journals and conferences related to her research topics. For additional information: http://www.iit.cnr.it/staff/elena.renda.

Paolo Santi received the Laura Degree summa cum laude and Ph.D. Degree in Computer Science from the University of Pisa in 1994 and 2000, respectively. He has been researcher at the Istituto di Informatica e Telematica - CNR in Pisa, Italy, since 2001, and Senior Researcher since 2009. During his career, he visited Georgia Institute of Technology in 2001, and Carnegie Mellon University in 2003. His research interests include fault-tolerant computing in multiprocessor systems (during Ph.D. studies), and, more recently, the investigation of fundamental properties of wireless multihop networks such as connectivity, topology control, lifetime, capacity, mobility modeling, and cooperation issues. He has contributed more than 50 papers and a book in the field of wireless ad hoc and sensor networking, he has been General Co-Chair of ACM VANET 2007 and ACM VANET 2008, he is Technical Program Co-Chair of IEEE WiMesh 2009, and he is involved in the organizational and technical program committee of several conferences in the field. Since February 2008, Dr. Santi is Associate Editor for IEEE Transactions on Mobile Computing. He is a senior member of ACM and SIGMOBILE. For additional information: http://www.iit.cnr.it/staff/paolo.santi.

Simone Burresi received the Laurea Degree in Computer Science from the University of Pisa in 2007. Currently he is an IT consultant for Value Team S.P.A., an Italian IT consultancy and services company of the Value Partners Group.