Sixth Annual IEEE International Conference on Pervasive Computing and Communications

MESHCHORD: A Location-Aware, Cross-Layer Specialization of Chord for Wireless Mesh Networks

Simone Burresi
Dept. of Computer Science
Univ. of Pisa, ITALY

Claudia Canali
IIT-CNR
Pisa, ITALY

M. Elena Renda
IIT-CNR
Pisa, ITALY

Paolo Santi
IIT-CNR
Pisa, ITALY

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 (called MESHCHORD*) 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. Through extensive packet-level simulations, we show that* 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.*

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 [14]), 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 [15] 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 application 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). Overall, the study reported in this paper suggests that MESHCHORD can be successfully utilized for implementing resource sharing applications in wireless mesh networks.

2 Related work and contribution

Recent papers have addressed the problem of enabling P2P resource sharing in mobile ad hoc networks (MANETs) [4, 7, 11, 13, 16]. 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. Approaches based on this idea are [2, 9, 10]. 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.

Only a few recent papers have explored how P2P ap-

0-7695-3113-X/08 $25.00 © 2008 IEEE
DOI 10.1109/PERCOM.2008.25

proaches can be applied to wireless mesh networks. In [1], the authors evaluate the gain that can be obtained from 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 [5], some of the authors of this paper introduced a two-tier architecture for file sharing in wireless mesh networks: 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 [5] under many respects. While the emphasis in [5] 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. Another major difference with respect to [5] is that we consider Chord [15] instead of Viceroy [8] 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. 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*. Finally, differently from [5], we investigate also the performance of information retrieval in terms of percentage of successful queries and query response time.

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 [2, 9, 10][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). 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.

[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.

Finally we want to cite [3], where the authors present a (packet-level) simulation-based investigation of Chord performance in MANETs environments. The main finding of [3] 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%.

The main difference between our study and the one reported in [3] is that we target mesh networks instead of MANETs, and that we also study a location-aware, cross-layer specialization of Chord for this type of networks. Contrary to what reported in [3], our analysis shows that Chord, and in particular our proposed specialization MESHCHORD, can successfully be applied in wireless mesh network scenarios.

3 MESHCHORD

3.1 Network architecture

Similarly to [5], we assume a two-tier architecture for file/resource sharing: the lower tier of the architecture is composed of (possibly) mobile mesh clients (clients for short), which provide the content to be shared in the P2P system; the upper tier of the architecture is composed of stationary mesh routers (routers for short), 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 to a router forming the DHT at the upper tier of the architecture.

We assume routers are stationary, but they can be switched on/off during network lifetime. 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, on the procedures needed to deal with client mobility, and to add/remove resources from the distributed index, see [5].

3.2 Basic Chord operations

The DHT approach investigated in this paper is Chord [15]. 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 1), i.e., peer p manages keys comprised between its own ID and the ID of the predecessor of p in the unit ring (denoted $range(p)$). Chord maps peer and resource IDs into the unit ring using a hashing function, named $Sha1$, which has the

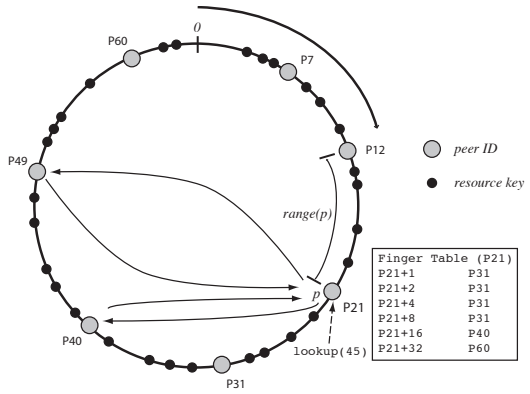

Figure 1. Basic Chord operations ($m = 6$).

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.

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 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 1). 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 1).

When a $lookup(x)$ operation is invoked at peer p and the operation cannot be resolved locally (because x is not within $range(p)$), a message is sent to the peer p' with largest ID $< x$ 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 $< x$ 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)[2]. Referring to Figure 1, 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 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. Optimal setting of this parameter 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 [5]):

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

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[3].

[2]This corresponds to the iterative method for implementing $lookup$ operations in Chord [15]. Also recursive $lookup$ implementation is considered in the original Chord design.

[3]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.

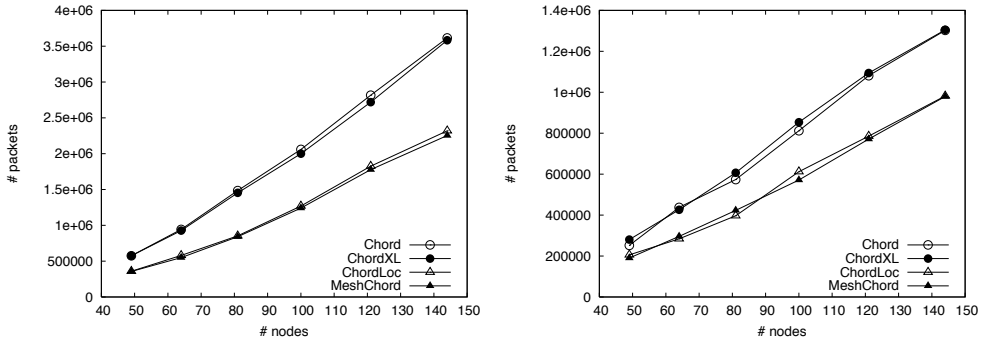

Figure 2. Total number of packets exchanged in grid networks for increasing number of n: almost static network (left) – $(1 - p_{leave}) = 0.999$, and very dynamic network (right) – $(1 - p_{leave}) = 0.9$.

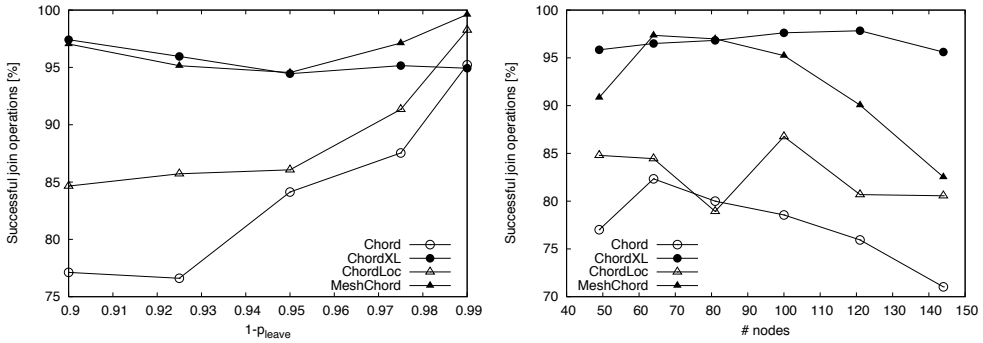

Figure 3. Percentage of successful join operations in grid networks for $n = 100$ and increasing values of $(1 - p_{leave})$ (left), and for increasing value of n with $(1 - p_{leave}) = 0.9$ (right).

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. More specifically, whenever a node 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.

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 [12]. 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).

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 DSR [6].

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}. 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

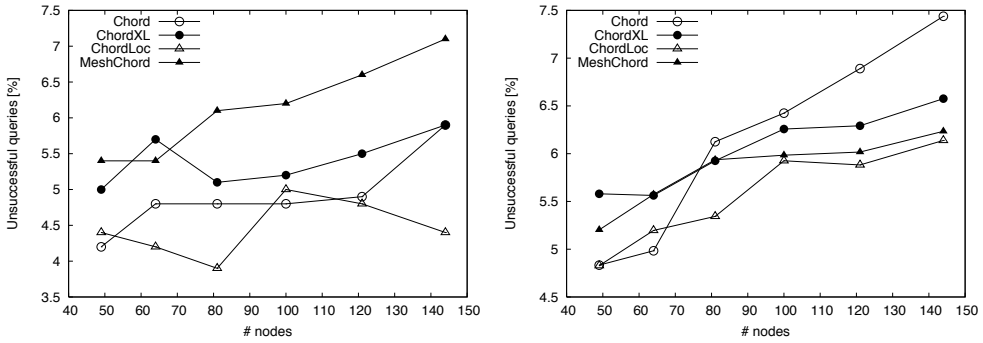

Figure 4. Percentage of unsuccessful queries in very dynamic networks for increasing values of n: grid topology (left), and random topology (right).

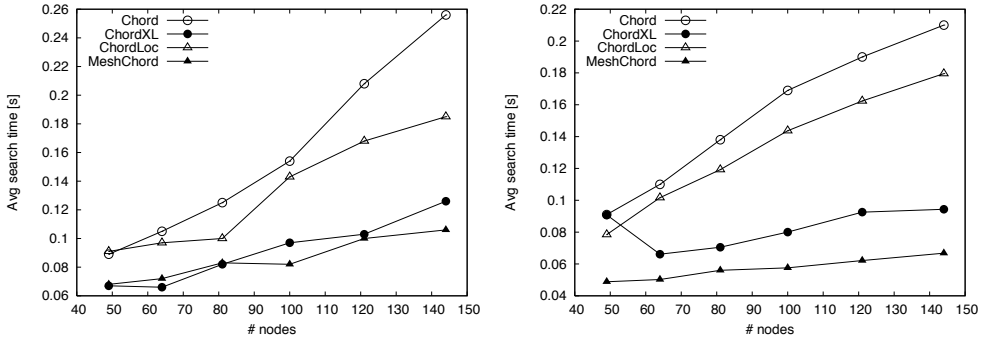

Figure 5. Average query response time in very dynamic networks for increasing values of n: grid topology (left), and random topology (right).

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.

When a peer u (re-)joins the network, it first has to find an active peer u' it is aware of (*hook peer*). To find the hook peer, before leaving the network u 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 1). 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. 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 two different sets of simulations, focusing on the effect of i) increasing the number n of peers, and ii) changing the number of join/leave events in the simulated time interval. 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[4].

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

[4]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.

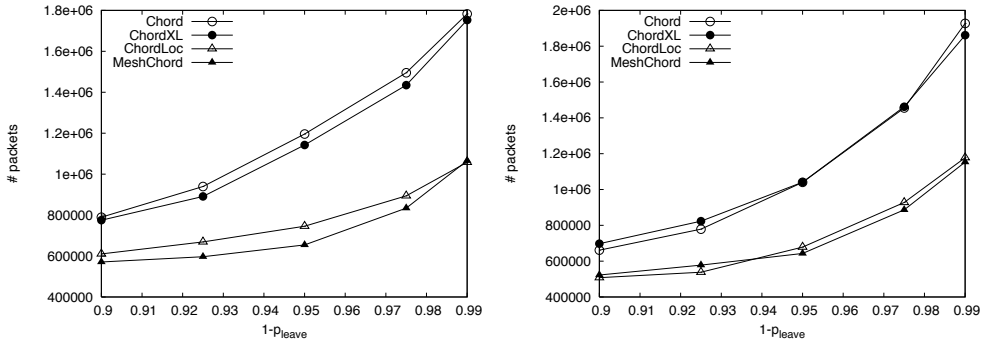

Figure 6. Total number of exchanged packets for different values of $(1 - p_{leave})$: grid topology (left), and random topology (right).

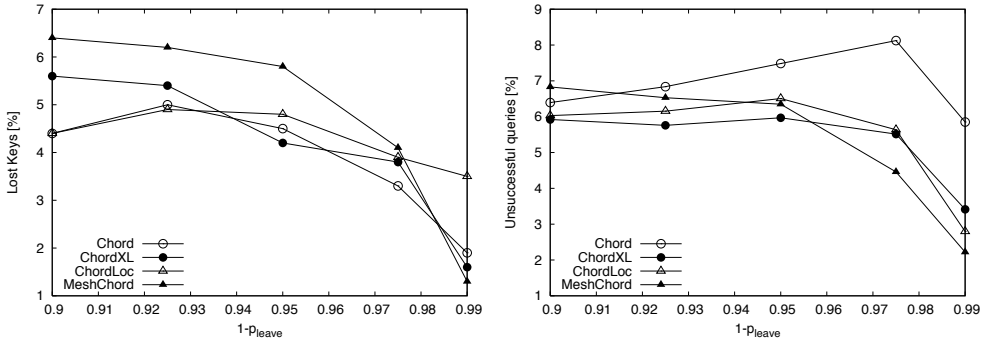

Figure 7. Percentage of unsuccessful queries for different values of $(1 - p_{leave})$: grid topology (left), and random topology (right).

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 due to lack of space, 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. Optimal settings of these parameters are: transmission range = $200m$, $\Delta = 200m$, stabilize interval = $7.5sec$.

4.2 Increasing network size

In the first set of experiments, we considered networks of size ranging from 49 to 144 peers (mesh routers). 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 $30sec$.

Figure 2 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 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 3. 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. Hence, the higher message exchange observed with cross-layering in dynamic networks is caused by the larger number of peers joining the Chord system 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 2), its effect on $lookup$ operations (which are at the basis of both new join and query resolution procedures) can actually be detrimental. In fact, 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 (hence, possibly corresponding to the last fingers in the finger table). This negative effect of location-awareness becomes more evident for larger networks (see Figure 3-right).

The results for the random topology scenario, not reported for lack of space, 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

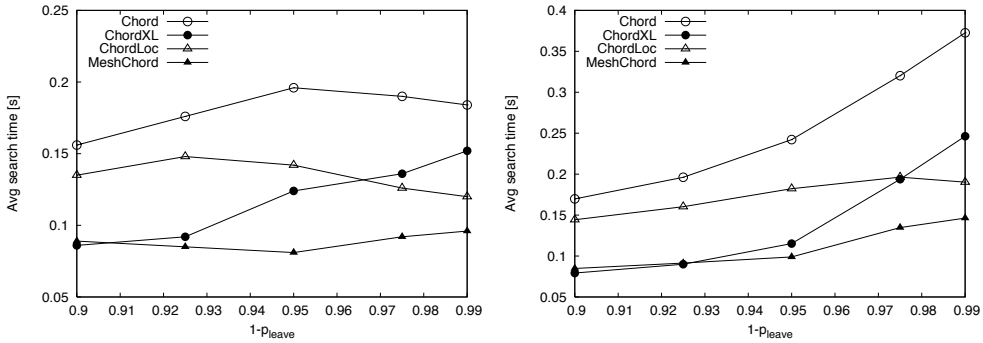

Figure 8. Average query response time for different values of $(1 - p_{leave})$: grid topology (left), and random topology (right).

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 4 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 is indeed caused by the fact that MESHCHORD tends to operate on a larger network (recall Figure 3), where the success rate is physiologically smaller. The results obtained in the almost stationary scenario (not reported for lack of space) showed an above 99% query success rate even for the basic Chord protocol.

Figure 5 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 stationary 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 6 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 2. 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 7 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.

Finally, Figure 8 shows that MESHCHORD is very effective in reducing the query response time with respect to Chord over the entire range of p_{leave} values considered.

5 Conclusions

The main finding of the study reported in this paper is that, contrary to what happens in MANET environments [3], the Chord approach can be successfully utilized for implementing file/resource sharing applications in wireless mesh networks. 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.

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 indeed lead to congestion when several applications are executed in the network concurrently with MESHCHORD. Thus, the problem of designing lightweight applications for file/resource sharing in wireless mesh networks is still open, and is matter of ongoing research.

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. Conti, E. Gregori, G. Turi, "A Cross-Layer Optimization of Gnutella for Mobile Ad Hoc Networks", *Proc. ACM MobiHoc*, May 2005.

[3] C. Cramer, T. Fuhrmann, "Performance Evaluation of Chord in Mobile Ad Hoc Networks", *Proc. ACM MobiShare*, pp. 48–53, 2006.

[4] 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.

[5] 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.

[6] D.B. Johnson, D.A. Maltz, "Dynamic Source Routing in Ad Hod Wireless Networks", *Mobile Computing*, n. 353, pp. 153–181, 1996.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] G. Riley, "The Georgia Tech Network Simulator," *ACM SIGCOMM MoMeTools Workshop,* 2003.

[13] F. Sailhan, V. Issarny, "Scalable Service Discovery for MANET", *Proc. IEEE PerCom*, 2005.

[14] `http://www.seattlewireless.net/`

[15] 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.

[16] O.Wolfson, B. Xu, H. Yin, H. Cao, "Search-and-Discover in Mobile P2P Network Databases", *Proc. IEEE ICDCS*, 2006.