Available online at www.sciencedirect.com
ScienceDirect
Pervasive and Mobile Computing 4 (2008) 62–91
pervasive and mobile computing
www.elsevier.com/locate/pmc

P2P multicast for pervasive ad hoc networks☆

Franca Delmastro*, Andrea Passarella, Marco Conti

CNR, IIT Institute, Via G. Moruzzi, 1 – 56124 Pisa, Italy

Received 19 December 2006; received in revised form 9 March 2007; accepted 23 March 2007
Available online 4 April 2007

Abstract

During the last few years, the proliferation of miniaturised devices with networking capabilities has provided the technological grounds for pervasive networking environments. It is not visionary to foresee a world of pervasive devices embedded in the environment interacting between them, and with those carried by users, via wireless communications. In addition, fostered by the diffusion of small-size, computational-rich mobile devices, the way content is generated, and accessed is changing with respect to the legacy-Internet paradigm. An ever-increasing share of the Internet content is generated directly by the users, and shared on the network (following the User-Generated Content model). While today the legacy Internet is still used to share user-generated content, it is reasonable to envision that pervasive networking technologies will represent the natural platform to support this new model. This will result in content being distributed on users' devices rather than on centralised servers on the Internet, and in users creating ad hoc networks to share content. The p2p paradigm is particularly suitable for this scenario, because communications will occur directly among users, instead of being necessarily mediated by centralised servers. Motivated by these remarks, in this work we focus on p2p *multicast* services over ad hoc networks aimed at sharing content among groups of users interested in the same topics. Specifically, starting from a reference solution in legacy wired networks (Scribe), we design a cross-layer optimised protocol (XScribe) that addresses most of the Scribe problems on ad hoc networks. XScribe exploits cross-layer interactions with a proactive routing protocol to manage group membership. Furthermore, it uses a lightweight, structureless approach to deliver data to group members. By jointly using experimental results and analytical models, we show that, with respect to Scribe, XScribe significantly reduces the packet loss and

☆ This work has been partly supported by the Italian Ministry for Research (MIUR) in the framework of the Project FIRB-PERF.

* Corresponding author.
E-mail addresses: franca.delmastro@iit.cnr.it (F. Delmastro), andrea.passarella@iit.cnr.it (A. Passarella), marco.conti@iit.cnr.it (M. Conti).

1574-1192/$ - see front matter
doi:10.1016/j.pmcj.2007.03.001

the delay experienced by multicast receivers, and increases the maximum throughput that can be delivered to multicast groups.

Keywords: Multi-hop ad hoc networks; P2p multicast; Cross-layer; Experimental evaluation; Modelling

1. Introduction

1.1. Background and motivations

The hardware and software progresses of the last ten years have provided the basic elements (wearable computers, wireless-network technologies, devices for sensing, and remote control, etc.) for the deployment of pervasive computing and communication systems. We are actually progressing towards Mark Weiser's vision, in which the environment is saturated with devices that have computing and communication capabilities, interacting among them and with the users. Besides this, the computing and networking capabilities of current mobile devices (PDAs, WiFi phones, etc.) are changing the way content is generated and accessed with respect to the legacy-Internet architecture (mostly based on the client-server model). Following the User-Generated Content model, users generate the content on their own, and share it over the network. The increasing popularity of services like YouTube, flickr, blogs, and grassroot journalism are just a few examples of this trend. Even though these services are currently implemented through the legacy-Internet architectures, a pure p2p communication model is more suitable to support this way of accessing information. Currently p2p systems are widespread on the Internet, and this trend will be even greater in pervasive networking environments. By progressing along these directions, we can envisage that in the ubiquitous Internet the content will be available everywhere, distributed among all networked devices, while the role of centralised Internet servers will diminish. The users will mine the network for relevant information and data, starting from nodes physically close to them.

This different method to access the information will have a strong impact on the network organisation. In a pervasive networking environment, the infrastructure-based wireless communication model will often not be adequate. Rather, self-organised networks will provide a more efficient and flexible solution for accessing network services and retrieving data from the network. We can thus envisage that the network structure will evolve towards a model in which dynamic, content-rich, ad hoc networking clouds will co-exist and be integrated with legacy-Internet backbones. A big share of the traffic will be generated and accessed via p2p communication models directly within the ad hoc networking clouds, without relying on the legacy infrastructure. Thus, since in this environment content management will be of primary importance, increasing attention is being devoted to design efficient p2p solutions for ad hoc networks. Specifically, during the last few years, there has been increasing interest in integrating legacy p2p systems and multi-hop ad hoc networks (see, for example, [1,2]).

One of the main obstacles to this integration is the fact that p2p systems are typically based on opposite assumptions with respect to those really held in ad hoc networks. Legacy p2p systems are designed to scale up to thousands of nodes. Furthermore, the networking

environment for which they are intended is typically resource rich, especially in terms of bandwidth. Thus, legacy p2p systems usually trade increased bandwidth consumption for higher scalability. Scalability to thousands of nodes is not a major issue for *flat* multi-hop ad hoc networks. Actually, theoretical and experimental results show that large-scale flat ad hoc networks are not very likely, because of intrinsic wireless capacity constraints [3,4]. Based on these observations [4] defines an "ad hoc horizon" for realistic flat ad hoc networks, consisting of 10–20 nodes, with peer-to-peer communications spanning 2 to 3 hops. Furthermore, ad hoc networks are definitely not a resource-rich environment, particularly in terms of bandwidth. Thus, trading increasing bandwidth consumption for greater scalability is not the best design choice. Experimental results show that, because of this mismatch, legacy p2p systems usually fail when used "as-they-are" in multi-hop ad hoc environments [5,6]. One of the main issues is therefore how to *efficiently* provide the same kind of p2p services implemented in legacy wired networks in multi-hop ad hoc networks also. Fortunately, previous studies show that p2p systems implementing DHTs can be designed to achieve high performance in ad hoc networks too (e.g. CrossROAD [7] and Virtual Ring Routing (VRR) [8]). However, a lot of work has still to be done to extend these results to more complex p2p platforms, providing further p2p services on top of DHTs.

One of the main lessons learnt from the previous work is that cross-layering is a fundamental approach to build efficient p2p systems in ad hoc networks [7,6]. To exploit this important result, in [9] we have defined an XL-CommonAPI that extends the original CommonAPI [10] to ad hoc environments. The original CommonAPI defines interfaces between components of a p2p system and applications, thus granting easy portability of p2p applications across different p2p systems' implementations. The extensions provided by the XL-CommonAPI are mainly focused on exporting fundamental routing- and DHT-level information to upper-layer services, thus allowing them to exploit complete knowledge about both the network topology and the overlay status, without requiring additional cross-layer interactions. The XL-CommonAPI is therefore a reference point to implement efficient p2p platforms on ad hoc networks, and to support new distributed services or optimise those already developed for wired networks.

1.2. Contributions

Starting from these considerations, in this paper we focus on p2p *multicast* services. Specifically, we consider p2p multicast protocols running on top of a p2p substrate providing a DHT (in Section 3 we discuss the reasons of this architectural choice). As a starting point, we choose a legacy solution made up of Pastry (at the DHT level), and Scribe (at the multicast level), since it is one of the most efficient solutions designed for wired networks [11]. After recalling the main features of Pastry and Scribe (Sections 3 and 4), we highlight the inefficiencies of such a p2p system when used as-it-is on ad hoc networks (Section 4.2). Based on these remarks, we propose a new solution that leverages cross-layer interactions between the p2p and the routing levels to optimise the multicast system implementation (see Fig. 1). Specifically, we replace Pastry with CrossROAD (whose main features are recalled in Section 3), and Scribe with XScribe (which are extensively described in Section 4). Since CrossROAD implements the XL-CommonAPI, and XScribe exploits cross-layer interactions via this interface, this is also an example of how to use the XL-CommonAPI to optimise p2p services for ad hoc networks.

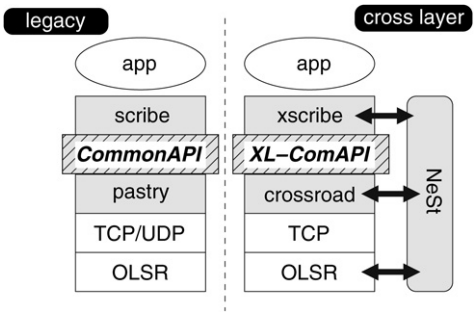


Fig. 1. Network architectures, emphasis on the p2p platforms and the cross-layer mechanisms.

While CrossROAD has already been proven to outperform Pastry on ad hoc networks (e.g., [5,6]), the main focus of this paper is the design and evaluation of XScribe. In Section 5, we evaluate the XScribe performance in comparison with Scribe, in terms of packet loss and delay. We report results from experiments run on a real multi-hop ad hoc network, implementing both p2p solutions. Results show that, in the majority of the cases, XScribe is able *at the same time* to halve the packet loss and the delay experienced by Scribe. To better understand p2p systems' behavior, in Section 6 we present an analytical model, which provides expressions for the *saturation throughput* of both Scribe and XScribe. The saturation throughput is defined as the maximum throughput that the multicast system is able to fairly deliver from groups' senders to groups' receivers. We exploit this model to show that XScribe is able to increase the scalability bounds of Scribe with respect to a number of parameters. Specifically, the proportional increase of the saturation throughput achieved by XScribe can be as high as 50%.

Despite its good performance, our results also suggest Xscribe's limitations and ways to further improve it. Specifically, XScribe implements a pure p2p delivery policy, in which no information about physical paths is exploited. We highlight that this is the main direction along which XScribe can be further improved, by blending together p2p multicast features and layer-3 multicast mechanisms.

2. Related work

Multicast support implemented at the p2p layer is just one of the available options proposed in the literature. Usually, multicast protocols are classified as operating at the network layer (layer-3), like routing protocols, or at the application layer, where 'application' denotes all possible layers above the transport.

Application-level multicast runs only at nodes involved in the related application, and it just requires standard unicast support from the routing level. The most recent proposals for wired environments run the multicast protocol on top of an overlay network based on a DHT [12–15]. This approach is very interesting for several reasons, as discussed in Section 3. The application-level multicast has been proposed also for ad hoc networks (e.g., ALMA [16] and PAST-DM [17]) even though, to the best of our knowledge, it has not yet been implemented on top of a DHT. This is mainly because efficient DHT implementations over ad hoc networks, that make such an approach feasible, have been

proposed just recently. Thanks to this, exploiting DHTs to support multicast services is an interesting direction to investigate in ad hoc networks also.

It should be pointed out that the application-level multicast potentially generates path stretch because just a subset of nodes can be used to deliver data. Moreover, although nodes that do not participate in multicast groups do not have to store multicast state, they have to forward data nevertheless. Therefore, it is not clear whether a multicast for ad hoc networks should be implemented at the network or at the application level. Examples of multicast protocols for ad hoc networks implemented at the network level are MAODV [18] and ODMRP [19] (other proposals, either implemented at the application or at the network level, are described in [20]). From this standpoint, there is actually an increasing trend towards blending together, in ad hoc networks' stacks, features usually implemented at the network and at the p2p levels (e.g., [8,7]). Therefore, an interesting solution could be a multicast system implemented within a routing layer that also includes p2p features. Results presented in this paper motivate this approach, and can be seen as a helpful intermediate step towards this goal.

The vast majority of multicast protocols for ad hoc networks proposed in the literature adopt structured approaches, i.e. the multicast protocol builds and maintains networking structures exclusively related to data delivery. For example, MAODV and ALMA use shared trees, while ODRMP and PAST-DM use meshes. A few others implement a structureless approach, in which the multicast protocol does not rely on any structure exclusively related to multicasting. Structureless multicasting requires that each sender is aware of at least a fraction of the receivers of the multicast group (we will refer to this property as *receiver awareness* in the following). Receiver awareness allows senders to directly deliver messages to receivers along the paths already built by the underlying routing protocol. Basically, receiver awareness replaces multicast structures. Examples of structureless multicast protocols for ad hoc networks are DDM [21] and RDG [22]. A structureless approach avoids the cost of maintaining another networking structure separated from the one already provided by the routing protocol. The main drawback of such an approach is clearly the cost of implementing receiver awareness (usually, receivers have to register with the senders, which then periodically poll them with alive messages).

XScribe actually follows a structure-less approach, since it exploits the network structure already defined by the underlying layers. With respect to DDM and RDG, XScribe works on top of a DHT and can straightforwardly support p2p applications. Furthermore, it implements receiver awareness in a more efficient way than periodic polling, and thus it better exploits the advantages of a structure-less design.

This paper complements our previous work on p2p multicast for multi-hop ad hoc networks. Specifically, in [23,24] we extensively evaluated the performance of Pastry and Scribe in ad hoc networks. In this work, we focus on the comparison between the legacy solution (Pastry & Scribe), and the cross-layer one (CrossROAD & XScribe). The main design features of XScribe have been presented in [25], together with a preliminary evaluation. In this paper, we provide a more complete evaluation, both from an experimental and an analytical standpoint. The subject of this work is comparing XScribe with Scribe, and we do not consider other multicast schemes proposed for ad hoc networks, such as ALMA, DDM, PAST-DM or ODRMP. This is because we are interested in multicast protocols implemented on top of a DHT. As discussed in Section 3, this is a

very promising direction not only for multicast over legacy wired networks, but also for multicast for ad hoc networks. As shown in [11], Scribe is the reference protocol for this particular approach. As a topic for future work, it would be interesting to compare XScribe with other application-level multicast schemes (e.g. ALMA). However, a fair comparison is not straightforward. Application-level multicasting usually defines a different overlay structure (typically in the form of a tree) for each multicast group. Instead, XScribe exploits the underlying overlay network (i.e., a DHT), for all the active multicast groups. A fair comparison will thus require us to take into account the overhead related to building a distinct tree for each group on the one hand, and the overhead related to DHT maintenance on the other hand. Finally, it would be also interesting to compare XScribe with the vast family of protocols designed for content-delivery networks. This is outside the scope of this paper, but it is definitely an interesting subject for future work.

3. DHTs for p2p multicast

Before describing the features of the p2p platforms shown in Fig. 1, it is worth briefly discussing why using a DHT to support p2p multicast is an interesting idea, and why using *structured* overlay networks is a reasonable choice. Using a DHT below the p2p multicast level is interesting for a number of reasons. Firstly, the task of defining a network structure that just encompasses the edge nodes is assigned to the DHT, and has not to be implemented by the multicast protocol itself. Secondly, the multicast protocol leverages the self-organising and self-recovery features of the DHT. Finally, the same DHT can be shared by several higher-level services running beside the multicast protocol.

Mainly designed for handling exact-match queries, a structured overlay (i.e. a DHT) is the most natural support for a p2p multicast protocol, in which single nodes have to send data to a well-defined set of receivers. In comparison with unstructured and hybrid overlays, structured overlays are usually considered (i) less efficient in terms of management, (ii) not able to exploit nodes' heterogeneity, and (iii) not able to support content-based queries. However, the discussion on these points is still open. For example, [26] shows that it is possible to address points (i)–(iii) by still using Pastry, and by achieving performance at least comparable to that provided by unstructured overlay networks. Thus, even though conceptually interesting, porting our analysis of p2p multicast protocols to unstructured and hybrid overlays is not the most compelling issue.

3.1. Pastry vs. crossROAD

Pastry defines a DHT using a logical circular address space where nodes and data are mapped. The logical address of a node is the hashed value of its IP address, while data are identified by key values. Thus, a key is associated to each message sent on the overlay to distribute or recover related data. Pastry delivers the message to the node whose logical id is the closest one to the hashed key. For the sake of scalability, each node keeps a partial view of the overlay network, i.e. it just knows about a limited set of nodes. The nodes that are kept in the set guarantee the forwarding correctness, i.e. that messages sent from anywhere eventually reach the correct destination, usually following a multi-hop path at the overlay level.

Therefore, the main costs of Pastry in terms of networking overhead are due to (i) the overlay creation and management that require periodic communications between nodes, and (ii) the multi-hop middleware routing caused by the incomplete knowledge of the overlay at each node, which possibly results in significant path stretches. These costs are well justified in large-scale wired networks, where the ability to scale to large number of nodes (possibly thousands) is correctly traded for additional bandwidth consumption (brought by points (i) and (ii) above). However, the cost of this trade-off is turned upside-down in multi-hop ad hoc networks, where the number of nodes is limited, bandwidth is scarce, and paths should be kept as short as possible since communications to nodes can be severely affected by unstable links.

CrossROAD has been designed to overcome Pastry's inefficiencies on multi-hop ad hoc networks. It provides the same DHT features, but implements them via a cross-layer optimised approach. Specifically, CrossROAD interacts with a *proactive* routing protocol via the cross-layer architecture proposed in [27]. As shown in Fig. 1, these interactions are mediated by the NeSt module, which provides standard interfaces for cross-layering, thus granting both performance optimisations and stacks manageability. A node wishing to join a CrossROAD overlay embeds few bytes into periodic routing advertisements, announcing its participation to a specific service. This information eventually reaches all the other nodes in the network through the proactive flooding of the routing protocol. Therefore, every node in the overlay knows the IP address of all the other nodes currently running the same service, and it is able to autonomously build the overlay by simply hashing their IP addresses. In this way, CrossROAD drastically reduces the bandwidth overhead with respect to Pastry [5,6], because building the overlay does not require any connection between nodes to exchange overlay information. Furthermore, p2p messages always travel just one-hop on the overlay because every node knows all the others (assuming that the routing protocol provides a consistent view at all the nodes). Even though using a proactive routing protocol over ad hoc networks might sound expensive, experimental results [5] showed that the overhead of the proactive routing protocol (OLSR in the particular case) is completely affordable. More details about CrossROAD operations can be found in [7,5].

Finally, note that CrossROAD exports to upper-layer services specific cross-layer information derived by its interactions with the routing protocol. This allows distributed services to optimise their behavior without requiring additional cross-layer interactions. However, in case a service needs a specific interaction with another layer of the stack, it can implement it through the general mechanism provided by the NeSt. In case of XScribe, cross-layer information derived by CrossROAD are used to distribute content on the overlay, following the main principles of DHTs. In addition, it defines an independent cross-layer interaction with the routing protocol to implement the membership management policy (see Section 4.3).

4. P2p multicast systems

4.1. Scribe

Scribe is a p2p shared-tree multicast protocol. It identifies each tree with a topic, and defines a root node for each topic as the node in the overlay whose address is the closest

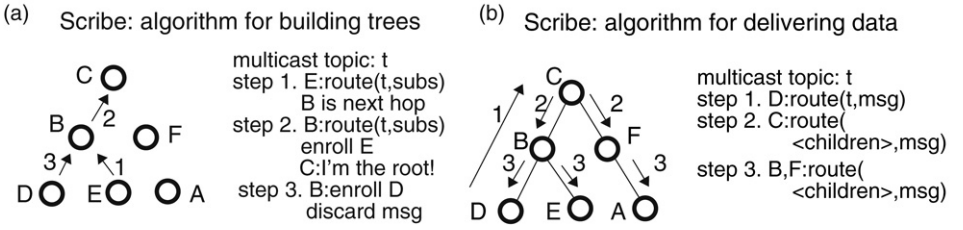


Fig. 2. Scribe tree construction (a), and data delivery (b).

one to the hashed topic. For example, in Fig. 2 the root is node C. Multicast trees are built through the well-known reverse path algorithm. Each node willing to join the tree sends a subscribe message specifying the topic as the key. An intermediate node (in the overlay path between the subscribing node and the root) that receives such a message, either subscribes itself to the same topic if it is *not* a member of the tree (e.g., node B after step 1 in Fig. 2(a)), or discards the message otherwise (e.g., node B in step 3 in Fig. 2(a)). In both cases, it enrolls the node from which it received the message as a child. Messages to be delivered over the tree are firstly sent towards the root of the topic (step 1 in Fig. 2(b)), and subsequently delivered by each parent to its children (steps 2 and 3 in Fig. 2(b)). Parent–child relationships are periodically refreshed through HeartBeat Messages sent by each parent to each child (application messages are also used as implicit HeartBeats). Upon missing a specified number of HeartBeats, a child assumes that the parent is no longer in the overlay, and sends a new subscribe message. This also allows the identification of the new root upon a failure or disconnection of the previous one. Children of the old root node detect that the root node is no longer there, and send subscribe messages that eventually arrive at the node currently closest to the topic, i.e. the new root. Nodes at lower levels in the tree are not affected by root failures. Finally, the current root node periodically checks whether it is still the node with the closest id to the topic. If it is not (e.g., because a new node joined the overlay), a new node has to become root. The old root sends a subscribe message to the new one, attaching the whole tree to the new root.

4.2. Scribe strengths and limitations

The design of Scribe has a number of interesting features. First of all, it allows the application to define multicast groups in a straightforward way by identifying groups with topics translated into logical ids in the DHT address space. Furthermore, when the network is stable enough, it minimizes the tree management traffic thanks to the implicit HeartBeats mechanism. Finally, it leverages the self-organising and self-healing features of the underlying DHT very effectively. The mechanisms described in the previous section to detect parent failures, recover the tree structure, and manage root changes guarantee loop freedom, and keeps recovery traffic local to the nodes actually involved in the failure detection and repair.

Despite these nice properties, Scribe also has features that are definitely not suitable for multi-hop ad hoc networks. Firstly, the data delivery mechanism is highly centralised. Messages have to first reach the root, and then are forwarded on the tree. Clearly, this is not efficient since it tends to saturate network resources around the root node, as shown by experiments in [28].

Secondly, Scribe uses a structured approach to multicast, which generally results in higher path stretches with respect to the minimum stretch granted by the DHT. Let us focus again on Fig. 2(b), and specifically on the path followed by messages sent by D to reach node E. Even though the mechanism could be optimised by avoiding messages being sent through the root node, messages from D to E have to pass through node B anyway, because they have to be delivered over the structure defined by the multicast protocol. Generally, it is easy to show that the best path on the overlay between D and E does not necessarily include B. Thus, such a delivery policy generally extends the physical path between senders and receivers more than what is strictly required by the DHT.

A third problem is also related to the structure defined by Scribe. Under dynamic or unstable conditions, the management traffic might grow significantly. Link instability, which is typical of wireless environments, may actually result in the disappearance of the temporaneous nodes from the overlay. This may increase the management traffic at the Scribe level to repair the multicast structure. In addition, if the link instability is caused by network congestion, such recovery mechanisms actually exacerbate the problem instead of contributing to fix it, and may result in tree partitions and isolation of the nodes [28].

Finally, the fourth problem we highlight is the fact that Scribe uses a *pure p2p* data delivery mechanism, in the sense that no information about the real paths taken by messages in the network is used. For example, when node B in Fig. 2 delivers a message to D and E, it generates two distinct copies of the message. If the paths between B and D, and between B and E, partially overlap, such a pure p2p delivery results in the unnecessary replication of message transmissions over the physical network.

4.3. XScribe

To cope with most of the problems highlighted in Section 4.2, we propose XScribe, which is a replacement for Scribe optimised through cross-layer interactions. XScribe still intentionally uses a pure p2p data delivery strategy, not addressing the fourth problem highlighted in the previous section. However, it is able to deal with the other Scribe limitations. XScribe is heavily inspired by *stateless, explicit* multicast approaches such as DDM [21] and RDG [22]. The main differences between these protocols and XScribe are that (i) XScribe is implemented at the p2p level, and therefore provides a more friendly support for applications with respect to standard layer-3 multicast, and (ii) its group membership policy is implemented via dedicated cross-layer interactions, drastically reducing the management traffic. For ease of explanation, we divide the XScribe operations into *data dissemination* and *membership management*, and discuss each aspect separately.

4.3.1. Data dissemination

To visually clarify how XScribe data dissemination works, Fig. 3 shows a simple case in which a multicast group is composed by 5 nodes, 2 of them generating data. Nodes are placed in their logical position in the CrossROAD ring overlay. Arrows represent flows of data between senders and receivers. As in DDM, in XScribe each sender of a multicast group is aware of all the other group members (we have defined this property as *receiver awareness*). Specifically, in XScribe each sender keeps a list with the logical addresses (in the overlay network) of the group members. Locally generated messages are disseminated in the group by sending a distinct message to each member of the overlay.

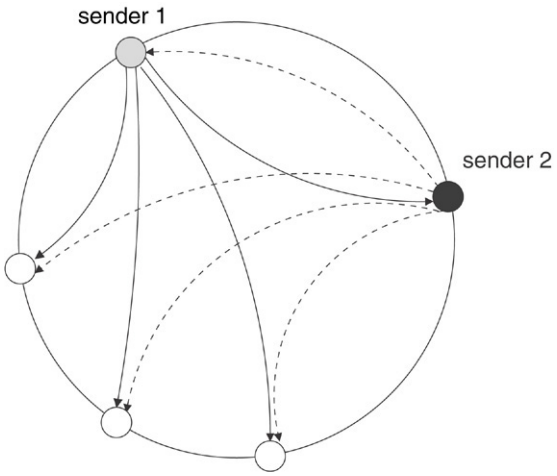


Fig. 3. XScribe data dissemination.

Thanks to receiver awareness, no structure related to multicast is required, and XScribe is thus a structure-less protocol. Clearly, the main issue of this approach is how to implement receiver awareness efficiently. The membership-management section addresses this aspect.

Even though this data-dissemination mechanism can be further optimised, it addresses the first, second, and third problems we have discussed in Section 4.2. The first advantage of the structure-less approach is to avoid centralisation, since no root node is defined anymore. In addition, XScribe also avoids the management traffic and path stretches brought into Scribe by the multicast structure. To understand this point, recall that in CrossROAD all the nodes in the overlay are aware of each other (thanks to the cross-layer interaction with a *proactive* routing protocol), and therefore they are just one hop away from each other *in the overlay*. In other words, no multi-hop *on the overlay* is required in CrossROAD. Therefore, if we consider any sender–receiver pair in XScribe, the sender can send messages to the receiver via a single-hop path *on the overlay*. Thus, messages are actually sent from a multicast sender to a receiver along the shortest (possibly multi-hop) path *at the network level*, according to the underlying layer-3 routing protocol.

Despite these advantages, it is easy to show that XScribe's dissemination policy could be further optimised. The repeated unicast used by XScribe is definitely a feature to be improved. However, note that in small-scale ad hoc networks Scribe tends to build two-level trees in which all the leaf nodes are direct children of the root node.[1] Thus, data dissemination from the root node occurs via repeated unicast in Scribe too. Despite this sub-optimal behavior, the results presented in the following sections show that XScribe is already able to significantly improve Scribe. Furthermore, the current version of XScribe allows us to stress the limits of a pure p2p delivery mechanism applied to multicast, and to understand how to further improve it (see Section 7 for a discussion on this point).

[1] This is essentially because in small-scale networks nodes tend to be aware of *all* the other nodes at the Pastry level.

4.3.2. Membership management

XScribe uses a cross-layer policy to manage group-membership, inspired by the main principle of CrossROAD. Even in this case, the main idea is to exploit the proactive routing traffic to spread information around. We assume that multicast groups can be mapped to positions in a *group bitmask*. One could envision a number of ways to define mappings, such as, for example, exploiting the fingerprint of the group id in a Bloom filter. XScribe maintains a group bitmask local to each node, which stores the multicast groups the node is subscribed to. As soon as the node subscribes to some groups (i.e. the bitmask is not completely cleared), the local bitmask is embedded into periodic messages generated by the routing protocol, and disseminated in the network. Further subscriptions/unsubscriptions are disseminated by setting/clearing the corresponding bit(s) in the bitmap embedded into routing packets. By inspecting received routing packets, each node in the network can be aware of all the members of the available multicast groups, which is exactly the information required by the data dissemination algorithm. A node is assumed to have ceased to be part of the network when no bitmasks of that node are received for a specified amount of time. Clearly, nodes that do not run the XScribe layer are not able to join multicast groups, while they are still able to run other applications based on CrossROAD.

Essentially, this is the same mechanism CrossROAD uses to advertise the presence of nodes in the overlay, and it is highly efficient from a bandwidth overhead standpoint. For example, if the bitmask size is 128 b, and the period used by the routing protocol to send updates is 2 s (the default value for Hello packets in OLSR), the overhead injected by a XScribe node is just 8 Bps. Furthermore, XScribe information does not need to be sent in separate frames at the MAC level, and this avoids additional accesses to the shared medium. This is very important whenever the network starts to become even slightly congested. Actually, experimental results related to CrossROAD (see [5]) show that the networking overhead to disseminate p2p-level information through the routing protocol is minimal. Since the additional information required by XScribe is very small, we do not quantitatively investigate this protocol's aspect, while we focus the performance evaluation on data delivery figures.

Finally, note that the membership management policy represents one of the main differences between XScribe and the other stateless multicast protocols such as DDM or RDG. Both DDM and RDG require each receiver to send a joint message to each sender of the group. Furthermore, they require periodic polling to refresh membership. These aspects of the protocols are usually neglected in evaluations, but they may represent a significant overhead.

5. Experimental evaluation

5.1. Methodology

To compare the legacy and the cross-layer multicast p2p systems we ran experiments in a real ad hoc network setup. Specifically, we used the Pastry and Scribe implementations provided by Rice University (FreePastry [29]) for the legacy system, while we implemented XScribe on top of CrossROAD. To have a realistic application environment,

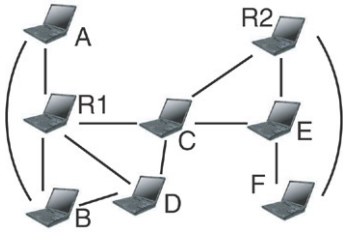
we also implemented a simple Whiteboard Application (WB) on top of both systems, allowing users to share drawings, writings, etc. Specifically, users run a Whiteboard instance on nodes of the ad hoc network, select a topic to join, and are then able to draw on a canvas, and receive drawings from the other users that selected the same topic. All the nodes whose users subscribe to the same topic form a multicast group.

To have a controllable and reproducible environment, in our tests WB was not run by humans, but by simulated users. Each user alternated between ON and OFF phases. During ON phases it drew strokes on the canvas, while during OFF phases it did nothing but receive other users' strokes. Both ON and OFF phases' lengths were exponentially distributed. An ON phase and the following OFF phase defined an active/idle cycle. Each trial was composed by 100 active/idle cycles, and each node running WB generated at least 500 messages.[2] To make trials start at the same time at different nodes, we synchronised the nodes before each trial, and scheduled the trial to start at the same time on each node. In the following, each trial configuration is identified by the application-load index, measured as the number of Packets Per Second (pps) generated by each user. This index is defined as the ratio between the average number of strokes generated in the ON phase of active/idle cycles, and the average duration of active/idle cycles. We found that this simple index is sufficient to correctly identify usage cases for our environments. The software implementing the protocol stacks can be downloaded from http://bruno1.iit.cnr.it/xscribe_exp/.

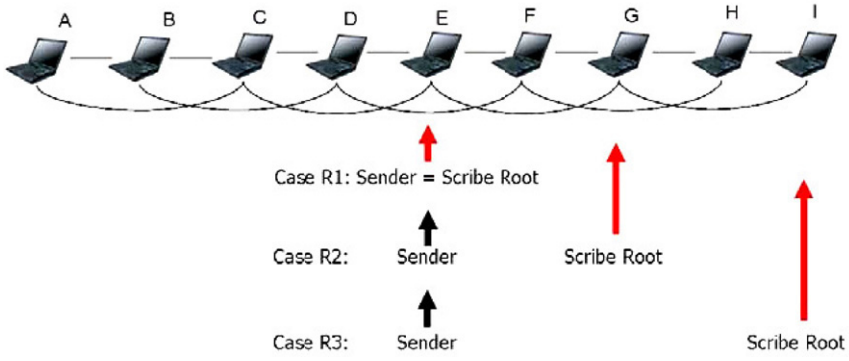
We characterise the architectures' performance at each node in terms of packet loss and delay statistics. Specifically, for any particular multicast group, the packet loss at node i is measured as $1 - \frac{R_i}{\sum_{j=1}^{N} S_j}$, where R_i is the number of messages received by node i related to that particular group, N the number of senders in the group, and S_j is the number of messages generated by the j-th sender of the group. Packet delays were measured by timestamping the transmission time at the sender, and the reception time at the receiver (recall that nodes were synchronised). Each configuration was replicated for 5 trials to have i.i.d. samples of the packet loss and delay statistics. Hereafter, we provide average values and confidence intervals of the performance figures for each configuration (specifying 90% confidence level, unless otherwise stated).

We have used the topologies shown in Fig. 4 to compare the performance of Scribe and XScribe. In both cases, all the nodes were IBM ThinkPad R50 laptops with integrated 802.11b wireless cards (Intel PRO-Wireless 2200). The OS was `linux-2.6.12.3`, loading the `ipw2200` driver for the network card. Actually, in both scenarios we used `iptables` to emulate multi hopping, and we cross-checked that paths were actually multi-hop by inspecting packet traces collected during the experiments. Although this methodology is unable to completely capture all the effects of wireless links' intricacies, it allowed us to closely approximate the system behavior in a realistic multi-hop setting, avoiding uncontrollable events due to the high variability of the wireless links' behavior. We did not run experiments in mobile conditions, in order to separate the effects of the different multicast architectures from those of nodes mobility. Correctly analysing these solutions

[2] A distinct message was sent for each stroke. The size of each WB message was 1448 B. In this way, each WB message resulted in a single frame of MTU size at the MAC level.



(a) Mesh topology.



(b) Rebundant chain topology.

Fig. 4. Topologies of the experimental testbed.

in mobile scenarios is a very interesting topic, but we have chosen not to address it in this work. When considering mobility, the performance differences between Scribe and XScribe are due to two distinct factors: (i) the different architectural choices, and (ii) the impact of mobility on the different architectures' features. To correctly understand the latter aspect it is first necessary to evaluate the performance differences of the alternative architectures alone (i.e., in static configurations), which is what we achieve in this work. This lays down the basis for a comprehensive understanding of Scribe and XScribe behavior in mobile configurations. Since the analysis in mobile configurations is a self-contained topic, and deserves a dedicated and carefully designed experimental planning, we consider it as one of the main subjects of future work.

The mesh topology (Fig. 4(a)) is representative of typical topologies one can expect to find in small-scale, general-purpose, self-organizing ad hoc networks. Nodes are not placed in a well-defined, ordered fashion (such as a chain or a grid), more than one path is available between any couple of nodes, and nodes physically close to each other may not be able to directly communicate (e.g., nodes D and F) due, for example, to the presence of obstacles such as concrete walls between them. Furthermore, in this topology, nodes A–E ran the whole protocol stacks, including the fact that in such a network nodes are likely to cooperate in forwarding tasks even though they may not be interested in the application data they are forwarding. We assumed that all the nodes running the WB application

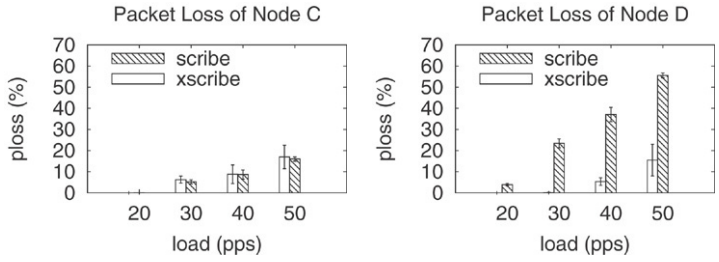
were interested in the same topic. Therefore, the multicast protocol worked with a single multicast group encompassing all the nodes. Node C was the root of the Scribe tree. Nodes C and D generated traffic at the application layer, thus acting as senders of the multicast group, while the other nodes only received application-level traffic.

As it is representative of general-purpose self-organizing ad hoc networks, we have used the mesh topology to understand the typical performance differences (in terms of packet loss and delay) between XScribe and Scribe. Instead, we used a redundant chain topology (Fig. 4(b)) to highlight one of the main architectural advantages of XScribe over Scribe, i.e. the fact that XScribe does not require a root node to manage the multicast group. To this end, we ran experiments in which just one node, placed at the center of the topology (node E in the figure), was sending messages to the group, while the root node was placed in different positions. Specifically, we defined three configurations (R1, R2, and R3) in which the Scribe root node was located on node E, G, and I respectively. The main purpose of the experiments run on this topology is to isolate the effect of the Scribe root node placement, and highlight the advantages of a solution like XScribe, that does not require any rendezvous point. To this end, we took particular care of avoiding effects related to routing instabilities (see the Appendix for a thorough discussion on this point). Therefore, we added a minimum path redundancy to the original chain topology, by allowing communications not only between adjacent nodes but also between nodes placed two positions apart in the chain. Furthermore, we ran experiments with a light traffic load (10 pps), to avoid congestion effects due to higher application loads. In each configuration, we measured the delay and packet loss achieved by Scribe and XScribe, averaged over all the nodes (again, the results we present hereafter are averaged over 5 replicas, and confidence intervals have a 90% confidence level). These experiments are partly overlapped with those presented in [23,24], in which we placed the Scribe root node at one edge of the mesh topology (Fig. 4(a)). However, those results are a particular "showcase" of what happens when the Scribe root node is not at the center of the topology. By using this purposely designed topology, in this work we quantitatively analyse the performance worsening of Scribe as a function of the physical distance between the root node and the other nodes in the multicast group, and especially from the sender node.[3]

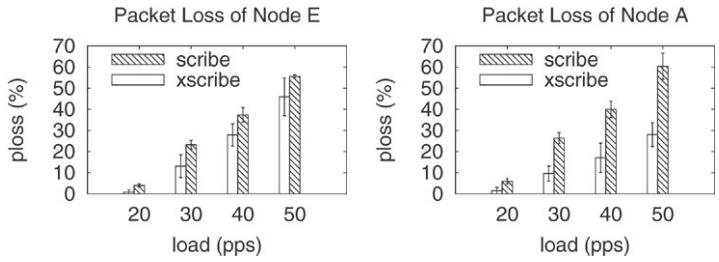
5.2. Experimental results in the mesh topology

In this section we deeply analyse the two performance indices, i.e. the packet loss (Section 5.2.1), and the delay (Section 5.2.2) achieved in the mesh topology. When inspecting the delay samples of both architectures, we found random and unexpected spikes. We found that these high values are exclusively due to routing-protocol misbehavior, and are equally likely running both Scribe and XScribe. Therefore, we decided to remove them from the performance evaluation. This allows us to evaluate the p2p multicast solutions in an optimistic case, in which the routing protocol behaves as expected, and we can thus fairly compare the architectural differences between Scribe and XScribe. The detailed explanation of these routing anomalies is presented in the Appendix.

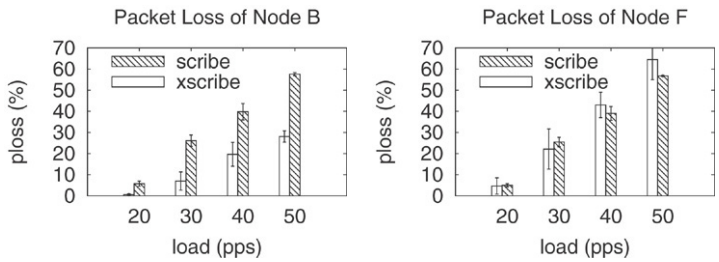
[3] Recall that in XScribe all the messages have firstly to be sent to the root node and then forwarded to all the receivers.



(a) Node C. (b) Node D.



(c) Node E. (d) Node A.



(e) Node B. (f) Node F.

Fig. 5. Packet loss at each node.

5.2.1. *Packet loss*

Fig. 5 shows the packet loss experienced by each node under the two alternative architectures as a function of the application load (loads below 20 pps resulted in no packet loss, and are thus omitted). Plots are presented starting from the center of the topology (node C), towards the edges. Since both architectures use TCP at the transport layer, one could expect not to see any packet loss. Actually, both Pastry and CrossROAD use internal queues (of the same size) to store messages going to be sent. Packet loss actually occurs when these queues fill up, and is thus a side effect of network congestion. In our setup we used the Pastry default value for the queue size, i.e., 128 messages. Properly dimensioning the queues to find the right balance between delay and packet loss depends on the particular application demands (actually, in other sets of experiments we have completely removed packet losses by allowing queues to grow to unlimited size).

Several interesting observations can be drawn from these plots. In general, it is evident that XScribe drastically reduces the packet loss with respect to Scribe. In more detail, at nodes D, A, and B the packet loss is always more than halved with respect to Scribe. At node E, the packet loss experienced by XScribe is lower than in the case of Scribe, but the reduction is less marked. Instead, different observations should be drawn with respect to node C (the Scribe root), and node F. Referring to node C, the packet loss experienced by the two systems is almost the same at the different traffic loads, and is reasonably low. Actually, this is expected. In fact, both in Scribe and XScribe, node C receives only packets from node D. Since the packet loss is mainly due to overflows at the sender's queue, the two systems experience the same performance. On the other hand, analysing the packet loss measured by node F, XScribe experiences slightly higher values only for high traffic loads (i.e. 40 and 50 pps). Note that in case of Scribe node F receives all the messages from a single TCP connection originated by node C, which is a 2-hop away. In the case of XScribe, node F receives the message over two concurrent TCP connections (from C and D), one of which spans 3 hops. Thus, at high traffic loads the longer connection suffers more than the others, and the sender's queue at node D fills up more quickly, thus causing a higher packet loss.

As a final remark, it should be noted that the packet loss we measured is also influenced by the routing anomalies discussed in the Appendix. In conjunction with these anomalies, route failures occur, and TCP retransmits failed packets until a new route is established. Thus, the senders' queues at the p2p level tend to fill up, and packet loss becomes more and more likely. Unfortunately, it is not possible to establish the exact share of packet loss related to these events. Therefore, our results might overestimate the packet loss once routing anomalies are fixed. However, since routing misbehavior is equally likely for Scribe and XScribe, the ranking among the two architectures that we have derived in this section is still valid.

In conclusion, the experimental results show that in the majority of the cases, XScribe performs better than Scribe, even though it generates more TCP connections. This means that, for any given usability threshold provided by the application in terms of maximum tolerable packet loss, XScribe allows the application to operate at higher loads than Scribe.

5.2.2. Delay

Table 1 shows the average values and the 99th percentiles of delays experienced by each node. We also associated with every single value its confidence interval. Three traffic loads have been selected as representative of light, medium, and high loads, and the experimental results are almost self-explanatory. When XScribe is used, the average delay is generally more than halved, while the 99th percentile is reduced by at least 1.4x, except for nodes E and F that experience higher delay percentiles in case of XScribe, but only for high traffic loads (50 pps). Coupled with the results related to packet loss, this means that, in the majority of the cases, for the same application load XScribe is able to drastically reduce the packet loss and, *at the same time*, to reduce the average delay.

As a final remark, it is worth pointing out that in this set of experiments, the root node of Scribe is located at the center of the topology, which minimizes the distance with all the other nodes. The next section is devoted to analyse the effect of different root node locations on the packet loss and delay performance figures.

Table 1
Delay statistics at each node (s)

Node	Load (pps)	Average		99th percentile	
		Scribe	XScribe	Scribe	XScribe
C	5	0.173 ± 0.048	0.108 ± 0.013	0.590 ± 0.178	0.432 ± 0.067
	20	1.03 ± 0.170	0.5 ± 0.054	3.05 ± 0.382	1.53 ± 0.246
	50	2.10 ± 0.084	1.64 ± 0.249	3.89 ± 0.122	4.28 ± 0.593
D	5	0.167 ± 0.09	0.095 ± 0.011	0.314 ± 0.055	0.432 ± 0.067
	20	1.38 ± 0.158	0.497 ± 0.055	4.07 ± 0.665	2.09 ± 0.389
	50	3.38 ± 0.653	1.81 ± 0.373	4.81 ± 1.38	4.38 ± 0.307
E	5	0.21 ± 0.058	0.113 ± 0.007	0.827 ± 0.204	0.432 ± 0.089
	20	1.89 ± 0.294	0.588 ± 0.042	6.00 ± 1.33	2.56 ± 0.669
	50	4.57 ± 0.087	1.88 ± 0.353	7.52 ± 0.379	7.72 ± 1.27
A	5	0.162 ± 0.079	0.111 ± 0.006	0.852 ± 0.197	0.385 ± 0.058
	20	2.06 ± 0.347	0.592 ± 0.055	6.32 ± 1.39	2.00 ± 0.206
	50	4.91 ± 0.348	2.01 ± 0.369	7.78 ± 0.357	5.42 ± 1.11
B	5	0.167 ± 0.10	0.079 ± 0.032	0.835 ± 0.206	0.322 ± 0.051
	20	2.03 ± 0.295	0.445 ± 0.110	6.28 ± 1.20	2.00 ± 0.333
	50	4.75 ± 0.057	1.92 ± 0.342	7.70 ± 0.405	5.75 ± 0.621
F	5	0.217 ± 0.063	0.176 ± 0.067	0.859 ± 0.252	0.514 ± 0.07
	20	2.01 ± 0.222	0.750 ± 0.088	6.25 ± 1.15	2.99 ± 0.551
	50	4.55 ± 0.30	4.08 ± 1.04	7.68 ± 0.369	9.34 ± 0.532

5.3. Experimental results in the redundant chain topology

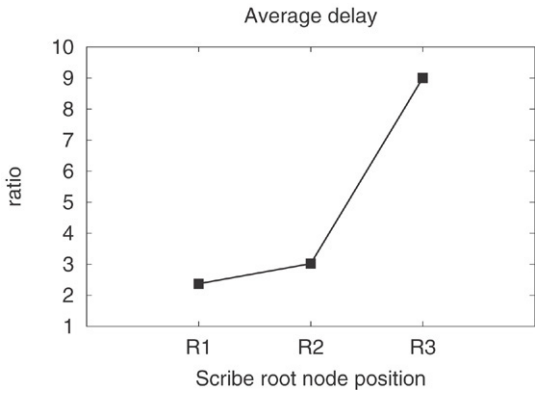
In this section, we quantify the Scribe performance worsening when the root node is moved from the center of the topology towards one edge. To this end, we ran three sets of experiments, in which the Scribe root was placed on node E, G, and I, respectively, while just the central node of the topology (E) was sending messages to the group. As shown in Fig. 4(b), we refer to these configurations as R1, R2, and R3, respectively. Note that XScribe does not require a root node, and therefore its performance is the same in the three configurations.

First of all, the performance in terms of packet loss is acceptable both for Scribe and XScribe. Specifically, all the nodes experience no packet loss when XScribe is used. In the case of Scribe, we measured non-negligible packet loss (i.e. about 10%) on node A, just in experiments where the root was located at the opposite edge of the chain (i.e., on node I). This was expected, since packet loss is a by-product of congestion at the root node (as explained in Section 5.2). The application load we have used is not high enough to saturate the TCP connections from the root to the other members, except for quite long paths.

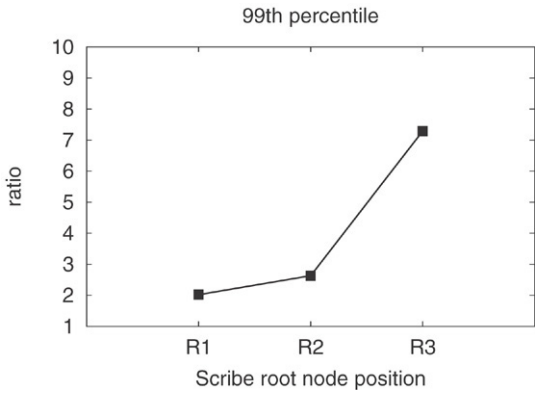
The impact of the Scribe root node position is evident when one focuses on the delay statistics. Specifically, Table 2 shows the average and the 99th percentile of the delay achieved in the different configurations by Scribe and XScribe. In detail, from each run we derived the average and 99th percentile of the delay achieved by every node, and we computed their mean value over all nodes. We report in the table the confidence intervals of the mean values, with a 90% confidence level. Basically, these results represent the

Table 2
Delay statistics in the redundant chain topology (ms)

XScribe		Average	99th percentile
		300 ± 4.45	1025 ± 35.6
Scribe	R1	710 ± 14.5	2071 ± 32.1
	R2	906 ± 34.2	2698 ± 180
	R3	2702 ± 377	7474 ± 852



(a) Average.



(b) 99th percentile.

Fig. 6. Ratio between the Scribe and XScribe delay statistics.

"average delay statistics" one can expect in a chain topology. To complement the results of Table 2, Fig. 6 shows the ratio between the Scribe and XScribe delay, both for the average value (a), and for the 99th percentile (b).

First of all, it should be noted that, even in the best Scribe configuration (R1), the average and 99th percentile of the delay are respectively 2.37 and 2.02 times the corresponding XScribe figures. It can be easily shown that in this case the application-level traffic

(i.e., WB messages) between the group members flows exactly along the same TCP connections both with Scribe and XScribe. Therefore, the XScribe performance improvement witnesses that the *lightweight architecture* of CrossROAD and XScribe results in lower application-level delay. Furthermore, Fig. 6 also shows that the Scribe delay statistics *exponentially* increase with the distance of the root from the center of the topology and from the sender node. Specifically, in an intermediate configuration (R2), the average and 99th percentile of the delay are respectively 3.02 and 2.63 times the corresponding XScribe figures. Finally, when root is on node I (R3), the Scribe average and 99th percentile delay grow up to 9.00 and 7.29 times the corresponding XScribe figures. Intuitively, the exponential increase of the Scribe delay is grounded in the increasing distance between the sender and the Scribe root, and in the fact that the throughput of the TCP connections between the root and the other group members decreases as $1/\sqrt{n}$, where n is the path length [3]. The exponential increase of the delay statistics show that Scribe may be able to support p2p applications in self-organized networks only in very particular configurations.

We would like to stress the fact that, even though these results are derived for a theoretical topology, they provide a general important result. Self-organized networks are totally unplanned, and the location of the root node is totally random. Having the root node at the center of the topology minimizing the distance from the other group members is thus quite a rare event. The typical performance difference between Scribe and XScribe in terms of delay is therefore represented by the R2 and R3 configurations. The experimental results discussed in this section confirm that XScribe is a much more suitable solution for general-purpose self-organizing networks with respect to Scribe.

6. Analytical evaluation of the XScribe improvements

To complement the results presented in Section 5 we now investigate in more detail the improvements that XScribe is able to provide in comparison with Scribe. Specifically, we present an analytical model providing the Scribe and XScribe *saturation throughput*, defined as the maximum application load generated by *each* sender that the multicast protocol is actually able to deliver to *all* the receivers. Loads higher than the saturation throughput can be delivered in general just to a *subset* of the receivers. Clearly, the higher the saturation throughput, the better. Comparing Scribe and XScribe in terms of saturation throughput actually complements the results we have achieved via experimental analysis. The comparison in terms of packet loss and delay presented before shows how XScribe is able to improve the QoS perceived by the users. The comparison in terms of saturation throughput, instead, shows to what extent XScribe is able to increase the region of the parameters' space for which the network is not overloaded.

6.1. Analytical model

For reasons of space, in this section we just present the main results of the analysis, i.e. the closed forms of the Scribe and XScribe saturation throughputs, hereafter referred to as γ_S and γ_{XS}, respectively.[4]

[4] The analytical proofs of these closed forms and other details on the model can be found in an extended version of this paper available at http://bruno1.iit.cnr.it/~andrea/docs/mcast_tr.pdf as a technical report.

Table 3
Model parameters

Symbol	Meaning
Γ	Saturation throughput of an 802.11 network whose nodes are in the same CS range
n	Number of nodes in the network
n_G	Average number of multicast groups in the network
n_S	Average number of senders in a multicast group
n_R	Average number of nodes, in a group, a sender has to send data to

First of all, let us clarify the main modeling assumptions. For fairness reasons, we impose that, in the saturation condition, all the senders of all the multicast groups generate the same application-level load. We assume that the average number of senders in any group is the same (n_S), and that, for any given group, each sender "sees" the same average number of receivers, n_R. In other words, each sender of any given group has to send data to n_R other nodes, on average. Please note that this also implies that the average size of multicast groups is $n_R + 1$.

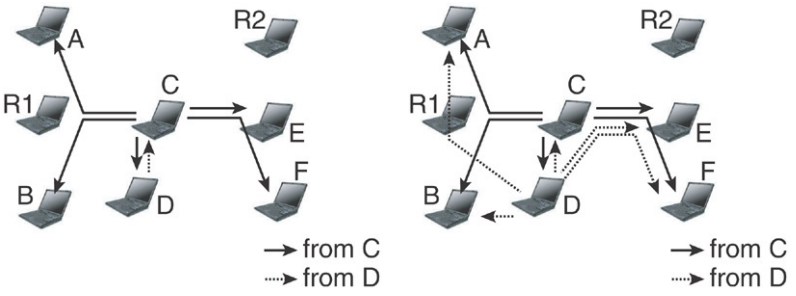
In the model, we also assume that all the nodes are inside the same Carrier-Sensing (CS) range. The CS range in 802.11 networks has proved to be much larger than the transmission range, especially at high transmission rates. For example, the CS range measured in [30] is 7 times larger than the transmission range at 11 Mbps, 3 times larger at 5.5 Mbps, 2 times larger at 2 Mbps, and 1.6 times larger at 1 Mbps. This means that nodes 7-hop away from each other (or even more) may be in the same CS range. As we mostly focus on multi-hop ad hoc networks within the ad hoc horizon (2–3 hops, 10–20 nodes), assuming that all the nodes are in the same CS range is reasonable. The last modeling assumption is that the overlay networks adopt UDP as the transport protocol. This is a reasonable choice for a number of applications that could run on top of p2p multicast systems, without necessarily requiring 100% reliability. For example, using UDP would provide reduced delays to the Whiteboard Application for an increased packet loss. This represents a reasonable trade-off. Furthermore, using UDP simplifies the analysis and, in our networking scenario, provides an upper bound for the saturation throughput achievable over TCP.

The Scribe and XScribe saturation throughputs are a function of a number of parameters, which are summarised in Table 3. Specifically, Γ denotes the saturation throughput of an 802.11 ad hoc network in which all the nodes are in the same CS range. Closed form expressions for this figure are available in the literature (the interested reader can refer, for example, to [31]). Furthermore, n denotes the number of nodes in the network, n_G the average number of multicast groups concurrently active in the network.

Having described the main model assumptions and parameters, we are now able to provide closed-form expressions of γ_S and γ_{XS}. Specifically, the XScribe saturation throughput (γ_{XS}) can be evaluated as

$$\Gamma = n_G \cdot \gamma_{XS} \cdot k\sqrt{n} \cdot n_R \cdot n_S \Rightarrow \gamma_{XS} = \frac{\Gamma}{n_G \cdot k\sqrt{n} \cdot n_R \cdot n_S}. \quad (1)$$

Eq. (1) is quite intuitive. In saturation's condition, the load generated by all senders of all groups has to meet the saturation load of the network, Γ. Following our assumptions, each



(a) Scribe. (b) XScribe.

Fig. 7. Example of flows' pattern.

sender generates γ_{XS} bps, which are sent to n_R receivers over a multi-hop path. Based on the well-known results in [32], the average length of a multi-hop path in an ad hoc network can be expressed as $k\sqrt{n}$, where k is a constant. Therefore, $\gamma_{XS} \cdot k\sqrt{n} \cdot n_R$ is the load generated on the whole network by a single sender. The expression in Eq. (1) can be easily derived based on this remark, and on the definitions of n_G and n_S.

Following a similar line of reasoning, it is possible to evaluate the Scribe saturation throughput (γ_S), as well. In this case, the derivation is less straightforward, as the cases in which the root node is or is not a sender have to be distinguished. Let us denote by h the number of groups for which the root node is a sender. It can be shown that the Scribe saturation throughput, for a fixed value of h, is

$$\gamma_S(h) = \frac{\Gamma}{k\sqrt{n} \cdot (n_G \cdot n_R \cdot n_S + n_G \cdot n_S - h)}. \tag{2}$$

It can also be shown that the probability of having h groups for which the root node is a sender (denoted by $p(h)$) is distributed according to a Binomial distribution with parameters n_G and $p \triangleq n_S/(n_R + 1)$. Therefore, the average value of the Scribe saturation throughput can be evaluated as follows[5]

$$\gamma_S = \sum_{h=0}^{n_G} \gamma_S(h) \cdot p(h). \tag{3}$$

6.2. Model validation

In order to validate our model, we ran a set of experiments on the mesh topology described in Section 5.2. Instead of using the real p2p systems, we replicated the set of UDP flows between the senders and the receivers that would have been generated by Pastry and CrossROAD to deliver the messages generated by Scribe and XScribe, respectively. Just as an example, Fig. 7 shows the set of flows in the case of Scribe and XScribe when

[5] Providing a closed form for Eq. (3) is not practical. However, in the technical report, we show that it is possible to evaluate the γ_S values by using the constructive method of Eq. (3).

Table 4
Validation parameters

Param	Value
k	0.67
n	8
n_R	5
n_S	2–6

the senders were nodes C and D. Traffic on the UDP flows was generated by using the `netperf` tool, version 2.4.1 [33]. Replacing the p2p systems with the equivalent set of UDP flows generated through `netperf` gives us a more controllable environment, allowing us to precisely investigate the scalability bounds of Scribe and XScribe nevertheless.

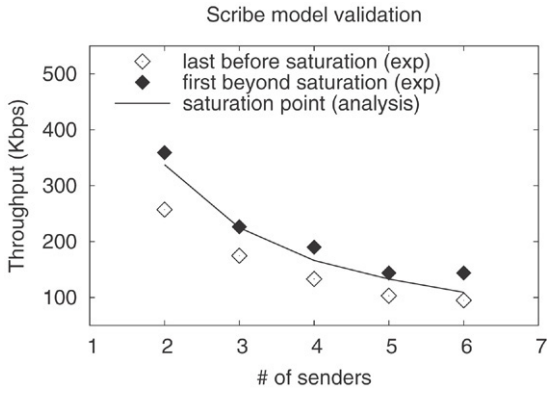
We tested the system under increasing numbers of senders, between 2 and 6 (the maximum possible in our mesh topology). In every case, we ran a set of experiments by increasing the load each sender generated on its flows, and we measured the throughput at the receivers. The configuration of an experiment is therefore defined by the number of senders, and by the load generated by each sender on each flow. We ran each configuration 10 times, each run lasting 100 s. Results presented hereafter are the average over the 10 runs. The 95% confidence interval was almost always within 2% of the average value, and was about 15% of the average value just in a few cases.

The outcome of an experiment (made up of 10 runs) was labeled as "good" if all the flows were able to correctly terminate the 10 runs. Under `netperf`, network overload manifests as a premature termination of some flows, which results in unavailability of the throughput measure at the receivers.[6] We labeled this type of experiment as "bad". Basically, "good" experiments represent cases in which the network was able to carry the load offered by all the senders, while "bad" experiments represent cases in which the network was overloaded (and the system was thus working beyond the saturation point).

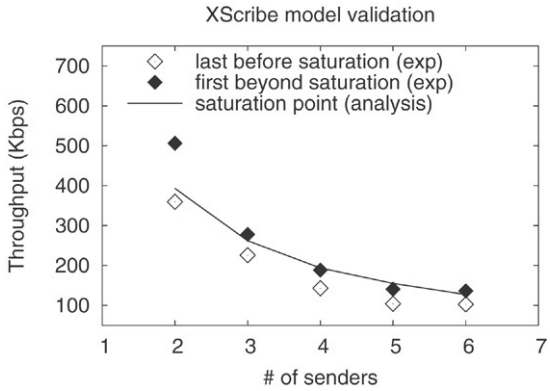
Fig. 8 shows the results from our testbed, compared with the saturation throughput predicted by the analytical model (i.e., by Eqs. (1) and (3)). Table 4 shows the model parameter values used for the validation. The value of k (where $k\sqrt{n}$ is the average path length) has been derived based on the average path length of the flows in all the configurations. Note that this value is remarkably aligned with the theoretical value of $2/3$ predicted by [32]. The value of Γ was derived exploiting the analytical model presented in [31], which provides the capacity (i.e., the maximum throughput) of an 802.11 network with a variable number of saturated nodes within a unique carrier-sensing region. Specifically, from the logs of the validation experiments, we derived, in each configuration (i.e., for each number of senders), the number of active nodes in the network, and we computed the analytical value of Γ corresponding to that number of active nodes.

In Fig. 8, for each number of senders, the white diamonds are related to the "highest-load good experiment", i.e. to the experiment with the highest offered load that resulted in a "good" experiment. Specifically, the white diamonds are the average values of the

[6] In detail, nodes experiencing a throughput below the offered load become unable to correctly manage the timers used by `netperf` to schedule message transmissions, thus resulting in a premature experiment termination.



(a) Scribe.



(b) XScribe.

Fig. 8. Model validation.

throughput measured at the receivers in the "highest-load good experiment". Likewise, the black diamonds are related to the "lowest-load bad experiment", i.e. to the experiment with the lowest offered load that resulted in a "bad" experiment. Recall that, due to `netperf` internals, in "bad" experiments only a subset of the receivers were able to correctly complete the experiment, and thus to provide a meaningful throughput value. The black diamonds are the average values of throughput measured at those receiver that correctly completed the "lowest-load bad experiment".

Clearly, for a particular number of senders, the real saturation throughput lies somewhere between the white and black diamonds. Fig. 8 actually validates our model, since the model predictions are, with a good degree of accuracy, within the region where the saturation throughput actually lies. Based on this result, in the following section we exploit the model to show that XScribe is able to significantly improve the saturation throughput with respect to Scribe, thus resulting in increased scalability with the load generated by applications.

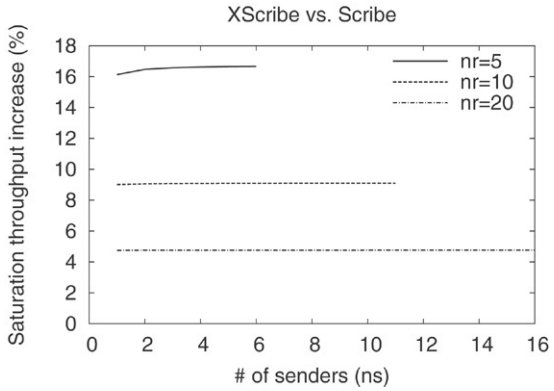


Fig. 9. XScribe improvement with respect to Scribe (ρ) as a function of n_S.

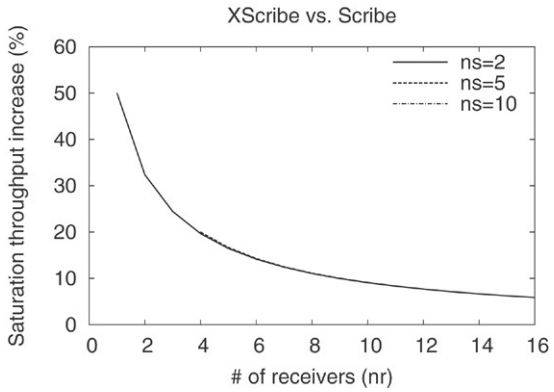


Fig. 10. XScribe improvement with respect to Scribe (ρ) as a function of n_R.

6.3. XScribe improvement in terms of scalability

To evaluate the XScribe improvement in terms of scalability, we focus on the proportional increase of the saturation throughput achieved by XScribe with respect to Scribe. This will be denoted by ρ, and can be clearly evaluated as $\rho = (\gamma_{XS} - \gamma_S)/\gamma_S$, where γ_{XS} and γ_S are defined by Eqs. (1) and (3), respectively.

First of all, it is easy to show that ρ does not depend on the multicast groups' popularity, i.e. the fraction of nodes that are members of multicast groups. In detail, we assume that the popularity is the same for all groups, and we define it as the ratio between the average group size and the average network size, i.e. $\alpha \triangleq \frac{n_R+1}{n}$. It is easy to show that both γ_S and γ_{XS} are proportional to $\sqrt{\alpha}$, and thus ρ is independent of α. Therefore, in the following we assume the same popularity of the experimental setup presented in Section 5, i.e. we fix $\alpha = 0.75$.

Figs. 9–11 plot ρ as a function of the other main system parameters, i.e. n_R, n_S, and n_G. Note that we limit the analysis to small-to-medium network sizes, i.e. we focus on n_R and n_S values below 20. This is coherent with the "ad hoc horizon" discussed in Section 1.

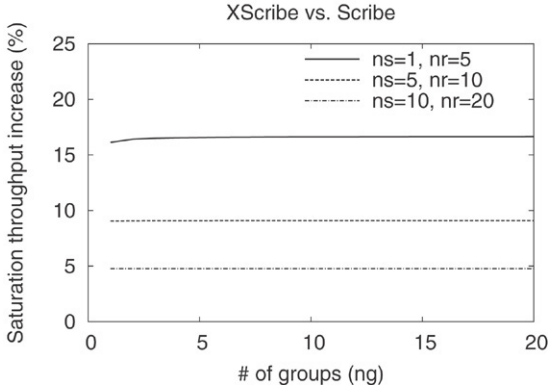


Fig. 11. XScribe improvement with respect to Scribe (ρ) as a function of n_G.

The proportional advantage of XScribe over Scribe is slightly dependent on the number of senders n_S (Fig. 9), and on the number of groups n_G (Fig. 11). This can be also derived directly by Eqs. (3) and (1). Specifically, γ_S is proportional both to $1/n_S$ and to $1/n_G$. Except for small n_S (n_G) values, γ_{XS} can be closely approximated with a function proportional to $1/n_S$ ($1/n_G$). Therefore, ρ is clearly independent of n_S (n_G), except for small n_S (n_G) values.

On the other hand, ρ is fairly sensitive to the n_R parameter, i.e. to the size of the group (Fig. 10). It is interesting to note that XScribe is able to increase the saturation throughput up to 50% with respect to Scribe. The advantage of using XScribe decreases as the average number of receivers seen by each sender grows, but it still remains non-negligible even for medium-size groups (i.e. for n_R between 10 and 20). The XScribe advantage is greater for small set of receivers, because, in general, the additional traffic generated by Scribe is due to carrying *all* multicast messages from the senders to the root before delivering them to the receivers. Clearly, this cost is less amortized, and Scribe performs worse, when the number of receivers is small.

In conclusion, the above analysis shows that the saturation throughput of both Scribe and XScribe decreases with the number of senders, receivers, and with the number of groups. This is because both γ_S and γ_{XS} can be closely approximated by functions proportional to $1/n_S$, $1/n_R$, and $1/n_G$. However, as shown by the ρ index, XScribe always outperforms Scribe, and is able to increase the saturation throughput up to 50%.

7. Conclusions and open issues

In this work we analysed the performance and limitations of p2p multicast systems for pervasive ad hoc networks. This is interesting because, thanks to the diffusion of resource-rich mobile devices, an ever-increasing share of the Internet content is actually generated by users themselves, and shared with other users over the network. Coupled with the increasing networking capabilities of mobile devices, it is reasonable to assume that a pervasive networking environment is made up of dynamic, wireless ad hoc networks, in which users generate and share content *within* the ad hoc networks, without necessarily

Fig. 12. Chain topology.

relying on centralized services over the Internet. Since the most likely communication paradigm aimed at this kind of sharing in the wired network is p2p, understanding how p2p systems should be designed to work over ad hoc networks is a hot topic. In addition, the increasing diffusion of user communities interested in common topics makes p2p multicast applications a suitable service in this environment.

In this work, we considered Scribe as a p2p multicast system, since it has shown good performance over legacy wired networks. However, experimental results demonstrated its major inefficiencies when it is used over ad hoc networks. Therefore, we presented and evaluated XScribe, which is a simple cross-layer replacement for Scribe, designed for multi-hop ad hoc networks. XScribe takes a structureless approach to multicast. Thus, it relies on the networking structure already defined by the underlying layers, instead of binding together multicast group members through network structures defined at the multicast level, such as trees or meshes. This is possible thanks to receiver awareness, i.e. in XScribe all the senders of a group know the group receivers. With respect to the other solutions exploiting receiver awareness, XScribe implements this feature via a very efficient cross-layer mechanism. Both the experimental results shown in Section 5 and the analysis discussed in Section 6 show that XScribe outperforms Scribe, thanks to these design features.

Despite these performance figures, there is still room for improvement. By design choice, XScribe adopts a pure p2p data delivery mechanism, in the sense that no information about the physical paths between senders and receivers is exploited during data delivery. This allowed us to stress the limits of such p2p delivery policies. The results in this paper clearly show that this policy actually limits the system scalability. Specifically, our analytical models show that, since a distinct message is delivered to each multicast receiver, the saturation throughput decreases as $1/n_R$, where n_R is the number of receivers "seen" by each sender. A possible fix for this is including cross-layer optimisations in the data-delivery phase as well. Specifically, possible portions of shared paths between senders and receivers could be efficiently exploited, since the same message could be transported just once along the shared portion of the path. Actually, this is a feature already implemented by traditional layer-3 multicast protocols. We think that integrating multicast features implementable at layer-3 (like sharing paths), and features related to the middleware layers (like exploiting subject-based routing) is the right balance to achieve efficiency while providing the applications with standard p2p multicast services.

Appendix. Analysis of routing misbehavior

Since experimental results obtained in the setup shown in Fig. 4 presented unexpected spikes in the delay samples, we decided to run some trials on a simple topology, to get an easier interpretation of the protocols' behavior. Specifically, we have used the chain topology shown in Fig. 12. In this topology all the nodes run the whole protocol stacks

(including the p2p systems and the Whiteboard application). A single multicast group encompasses all the nodes. Node C is the only sender, and all the other nodes are receivers. In this setup, the behaviours of Scribe and XScribe are remarkably similar. In the case of Scribe, since the root node is the only sender, it does not represent a centralisation point, since it does not receive traffic to be forwarded. Furthermore, the traffic pattern generated by Scribe and XScribe to deliver application-level data to the receivers is the same (i.e. data generated by C are delivered to each receiver on a dedicated TCP connection in both cases). Actually, since XScribe adds group-membership information to routing packets, the routing protocol behaves slightly different in the two cases, but the effects of its anomalies are very similar. For these reasons, hereafter we only discuss results from XScribe experiments that better highlight the anomalies we have found. Specifically, we discuss a single trial in which node C generates 20 pps (similar remarks apply also to the other configurations we used).

In this specific experiment, nodes A, B, and D measure no packet loss, while nodes E and F measure 37% and 40% packet losses, respectively. The log files show that node F loses *all* the routes for a few seconds during the trial, while in the same period node E has only a valid route to F (while the routes to all the other nodes have disappeared). At the same time, all the other nodes (i.e. from C to B) lose the routes to both E and F. This is not so surprising in this kind of scenario, since it is well known that in a multi-hop chain topology the loss of some Hello packets can cause a network partition.

Unfortunately, such route failures cause spikes of application messages delays. In conjuction with these events, nodes E and F experience delays in the order of 20 s, while the other nodes experience delays no greater than 5 s throughout the whole experiment. It is quite surprising that such route losses, and the associated delay spikes, also occur at very light traffic loads (5 pps), i.e. when the network is not congested. This suggests that they could be caused by some misbehavior of the routing protocol. To investigate this hypothesis, we deeply analysed the routing behaviour directly examining the packets sent on the network through the `tcpdump` sniffer.

According to the standard OLSR specification [34], Hello messages should be sent every 2 s, and TC messages (aggregating the link state of several nodes) every 5 s, or upon a topology change. Links advertised by Hello and TC messages are valid for the next 6 and 15 s, respectively. If they are not refreshed within these validity timeouts, they are removed from the routing table. Referring to the OLSR implementation we used, we noticed that, even though Hello and TC messages are generated on time by the OLSR process (i.e. precisely every 2 and 5 s), they experience significant delays either in being actually sent, or in being processed at the receiving side. Such delays occur because (i) this specific OLSR implementation aggregates routing messages in single packets to reduce the network load, and (ii) routing packets are not prioritized at the MAC level, and are thus queued together with application-level packets (especially at high traffic loads).

The delayed sending of routing packets thus makes the entire system more susceptible to route failures. Maintaining the original values of route validity timeouts, it can be sufficient to lose one routing packet, or receive it with a delay higher than the timeout, to cause a route failure. In our specific trial, node F lost a Hello packet sent by node E, and received the next packet 10 s after the last one correctly received. This caused the expiration of the validity timeout in F's routing table for the link with E, and the consequent route failures.

Similar events have been noticed at node E, i.e. it lost a Hello from B, and all the related routes. This example highlights how the additional delays of routing messages weaken the protocol reaction to packet loss.

Such misbehavior partly depends on the particular routing protocol implementation, but is highly determined by the fact that the routing traffic is not prioritized at the MAC level. As mentioned before, we decided to eliminate from our logs spikes in application message delays that occur because of these routing anomalies. This allows us to evaluate the p2p multicast solutions in an optimistic case, in which the routing protocol behaves as expected, and we can fairly compare the architectural differences between Scribe and XScribe.

References

[1] M. Gerla, C. Lindemann, A. Rowstron, p2p MANETs—New research issues. Available online: http://drops.dagstuhl.de/portals/index.php?semnr=05152.
[2] C. Lindemann, A. Rowstron (Eds.), ACM MobiCom MobiShare Workshop, September 2006. Available online: http://www.mobishare.org.
[3] P. Gupta, P. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory 46 (2) (2000) 388–404.
[4] P. Gunningberg, H. Lundgren, E. Nordstroem, C. Tschudin, Lessons from experimental MANET research, Ad Hoc Networks Journal 3 (2) (2005) 221–233.
[5] E. Borgia, M. Conti, F. Delmastro, MobileMAN: Design, integration and experimentation of cross-layer mobile multi-hop ad hoc networks, IEEE Communication Magazine, Ad Hoc & Sensor Network series 44 (7) (2006).
[6] M. Conti, F. Delmastro, G. Turi, Peer-to-peer computing in mobile ad hoc networks, in: P. Bellavista, A. Corradi (Eds.), The Handbook of Mobile Middleware, Chapman & Hall/CRC Press, 2006.
[7] F. Delmastro, From pastry to crossROAD: Cross-layer ring overlay for AD hoc networks, in: Proc. of IEEE PerCom MP2P Workshop, Kauai Island, Hawaii, March 2005.
[8] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, A. Rowstron, Virtual ring routing: Network routing inspired by DHTs, in: Proc. of ACM SIGCOMM, Pisa, Italy, September 11–15, 2006.
[9] F. Delmastro, M. Conti, E. Gregori, P2P Common API for structured overlay networks: A cross-layer extension, in: Proc. of MDC'06 Workshop, Niagara Falls, Buffalo, NY, June 2006.
[10] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, I. Stoica, Towards a common API for structured Peer-to-Peer overlays, in: Proc. of the the 2nd International Workshop on Peer-to-peer Systems, IPTPS'03, Berkeley, CA, February 2003.
[11] M. Castro, M.B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H Wang, A. Wolman, An evaluation of scalable application-level multicast built using peer-to-peer overlays, in: Proc. of INFOCOM, San Francisco, CA, April 2003.
[12] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron, SCRIBE: A large-scale and decentralised application-level multicast infrastructure, IEEE Journal on Selected Areas in Communications 20 (8) (2002).
[13] S. Ratnasamy, M. Handley, R. Karp, S Shenker, Application-Level multicast using content-addressable networks, in: 3rd International Workshop on Networked Group Communication, November 2001.
[14] S.Q. Zhuang, B.Y. Zhang, A.D. Joseph, R.H. Katz, J.D. Kubiatowicz, Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination, in: Eleventh International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV, 2001.
[15] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, SplitStream: High bandwidth multicast in cooperative environments, in: ACM SOSP, October 2003.
[16] M. Ge, S.V. Krishnamurthy, M. Faloutsos, Application versus network layer multicasting in Ad hoc networks: The ALMA routing protocol, Elsevier Ad Hoc Networks Journal 4 (2) (2006) 283–300.
[17] C. Gui, P. Mohapatra, Overlay multicast for MANETs using dynamic virtual mesh, Wireless Networks 13 (1) (2007) 77–91, doi: http://dx.doi.org/10.1007/s11276-006-1056-4.
[18] E. Royer, C. Perkins, Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing, 2000. Available online: http://www3.ietf.org/proceedings/00jul/I-D/manet-maodv-00.txt.

[19] S.-J. Lee, W. Su, M. Gerla, On-demand multicast routing protocol (ODMRP) for Ad Hoc networks, 2000. Available online: http://www.cs.ucla.edu/NRL/wireless/PAPER/draft-ietf-manet-odmrp-02.txt.
[20] S. Yang, J. Wu, in: Y. Xiao, Y. Pan (Eds.), New Technologies of Multicasting in MANETS, Nova Publishers, 2005.
[21] L. Ji, M. Corson, Explicit multicasting for mobile ad hoc networks, Mobile Networks and Applications 8 (2003) 525–549.
[22] J. Luo, P. Eugster, J.-P. Hubaux, Probabilistic reliable multicast in ad hoc networks, Ad Hoc Networks 2 (2004) 369–386.
[23] F. Delmastro, A. Passarella, M. Conti, Experimental analysis of p2p shared-tree multicast on MANETs: The case of scribe, in: Proc. of the Fifth IFIP Annual Mediterranean Ad Hoc Networking Workshop, MedHocNet 2006, Lipari, Italy, June 2006.
[24] A. Passarella, F. Delmastro, Usability of legacy p2p multicast in multi-hop ad hoc networks: An experimental study, EURASIP Journal on Wireless Communications and Networking 2007 (2007) 13 pages, doi: http://dx.doi.org/10.1155/2007/62089.
[25] A. Passarella, F. Delmastro, M. Conti, XScribe: A Stateless, cross-layer approach to p2p multicast in multi-hop ad hoc networks, in: Proc. of ACM MobiShare, September 2006.
[26] M. Castro, M. Costa, A. Rowstron, Debunking some myths about structured and unstructured overlays, in: Proc. of NSDI, May 2–5, 2005.
[27] M. Conti, G. Maselli, G. Turi, S. Giordano, Cross layering in mobile ad hoc network design, IEEE Computer (2004).
[28] F. Delmastro, A. Passarella, An experimental study of p2p group-communication applications in real-world MANETs, in: Proc. of IEEE ICPS REALMAN 2005 Workshop, Santorini, Greece, July 2005.
[29] FreePastry, Rice University. Available online: http://freepastry.rice.edu.
[30] G. Anastasi, E. Borgia, M. Conti, E. Gregori, A. Passarella, Understanding the real behavior of Mote and 802.11 ad hoc networks: An experimental approach, Pervasive and Mobile Computing 1 (2) (2005) 237–256.
[31] G. Bianchi, Performance Analysis of the IEEE 802.11 Distributed Coordination Function, vol. 18, no. 9, 2000 pp. 1787–1800.
[32] J. Li, C. Blake, D.S.D. Couto, H.I. Lee, R. Morris, Capacity of ad hoc wireless networks, in: Proc. of ACM MobiCom, 2001.
[33] Netperf traffic generator. Available online: http://www.netperf.org/netoerf/NetperfPage.html.
[34] Optimized Link State Routing Protocol (OLSR): RFC3626. Available online: http://www.ietf.org/rfc/rfc3626.txt.

Franca Delmastro is a Research Associate at the IIT Institute of the National Research Council (CNR), Italy. She received the M.S. Degree in Computer Engineering and the Ph.D. in Information Engineering, both from the University of Pisa, in 2002 and 2006 respectively. She joint IIT in 2003. Her research interests include ad hoc, mesh and opportunistic networks, with particular attention to routing and forwarding protocols, mobile p2p systems, multicasting for ad hoc networks, and distributed applications.

Andrea Passarella is a Researcher at the IIT Institute of the National Research Council (CNR), Italy. Before joining IIT, he was a Research Associate at the Computer Laboratory of the University of Cambridge, UK. He received the Ph.D. and M.S. Degrees in Computer Engineering, both from the University of Pisa, Italy, in 2005 and 2001, respectively. His current research is mostly on opportunistic and delay-tolerant networking. More generally, he works on ad hoc and sensor networks, specifically on p2p systems, multicasting, transport protocols, and energy-efficient protocols. His research interests also include mesh networks and wireless access to the Internet. He is Co-Editor of the book "Multi-hop Ad hoc Networks: From Theory to Reality" (Nova Science, 2007). He was TPC Vice-Chair for IEEE REALMAN 2005, ACM REALMAN 2006, and IEEE MDC 2006. He served and is currently serving in the TPC of several international conferences, including IEEE PerCom 2006–07 and IEEE WoWMoM 2006–07, and workshops. He is an Associate Technical Editor for IEEE Communications Magazine. He is a member of ACM SIGMOBILE.

Marco Conti is a research director at IIT, an institute of the Italian National Research Council (CNR). He co-authored the book "Metropolitan Area Networks" (Springer, 1997) and is co-editor of the book "Mobile Ad Hoc Networking" (IEEE-Wiley 2004). He has published in journals and conference proceedings more than 180 research papers related to the design, modeling, and performance evaluation of computer-network architectures and protocols. He served as the TPC chair of IEEE PerCom 2006, and of the IFIP-TC6 Conferences "Networking2002" and "PWC2003", and as TPC co-chair of ACM WoWMoM 2002, WiOpt '04, IEEE WoWMoM 2005, and ACM MobiHoc 2006. He also served as general co-chair of IEEE WoWMoM 2006 and as general chair of ACM REALMAN 2006. Currently, he is serving as general chair of IEEE MASS 2007. He is Associate Editor of Pervasive and Mobile Computing Journal, and he is on the editorial board of: IEEE Transactions on Mobile Computing, Ad Hoc Networks journal and Wireless Ad Hoc and Sensor Networks: An International Journal.