Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2007, Article ID 62089, 13 pages
doi:10.1155/2007/62089

Research Article

Usability of Legacy p2p Multicast in Multihop Ad Hoc Networks: An Experimental Study

Andrea Passarella and Franca Delmastro

Pervasive Computing and Networking Laboratory (PerLab), Institute for Informatics and Telematics (IIT), Italian National Research Council, Via G. Moruzzi 1, 56124 Pisa, Italy

Received 14 July 2006; Revised 16 October 2006; Accepted 27 November 2006

Recommended by Marco Conti

There has recently been an increasing interest in convergence of p2p and ad hoc network research. Actually, p2p systems and multihop ad hoc networks share similar features, such as self-organisation, decentralisation, self-healing, and so forth. It is thus interesting to understand if p2p systems designed for the wired Internet are suitable also for ad hoc networks and, if they are not, in which direction they should be improved. In this paper, we report our experience in running p2p applications in real multihop ad hoc network testbeds. Specifically, we used group-communication applications that require p2p systems made up of an overlay network and a p2p multicast protocol. In this paper, we present experimental results specifically related to the performance of a well-known p2p shared-tree multicast protocol (Scribe). Our results show that such a solution is far from being efficient on ad hoc networks. We emphasize that the *structured* multicast approach is one of the main causes of inefficiency, and suggest that *stateless* solutions could be preferable.

1. INTRODUCTION

The integration of p2p systems into multihop ad hoc networks is a very interesting and challenging topic that is attracting increasing attention [1, 2]. Actually, p2p systems and ad hoc networks share a number of similar features. P2p systems provide a decentralised, self-organising, and self-healing environment, along with features like subject-based routing, distributed data storage/retrieval, and load balancing. Such features, originally devised for p2p overlay networks in the legacy Internet, are also well suited for a decentralised and dynamic environment like a multihop ad hoc network. It is thus extremely important to understand how p2p systems designed for the Internet could be integrated into ad hoc networks.

Our approach to this study is twofold. Firstly, we run experiments on a real testbed, so as to take into account all the wireless links complexities that real systems have to deal with. Secondly, we deploy in our testbed a realistic prototype that includes all layers of the stack, from the physical up to the application one. Specifically, we have developed a distributed whiteboard (WB) at the application layer. WB allows users to share content within members of a community. It is an instance of group-communication (GC) applications that are an interesting example of p2p applications oriented to ad hoc networks (see Section 3.1).

We believe that both of these features are important to achieve clear understanding of the p2p system behavior in ad hoc networks. Actually, most of the research on multihop ad hoc networks has privileged a simulation and/or theoretical-driven approach. Simulation and theoretical analysis are very helpful tools since they allow to manage large scales, control parameters' variation, and so forth. However, propagation over wireless medium is so complicated to precisely model that experimenting on real testbeds is a must. Just relying on simulation and theoretical models might lead to conclusions that do not match real measures [3, 4]. Furthermore, little effort has been devoted to analyse the behavior of multihop ad hoc networks with respect to realistic applications. Much effort has been spent on evaluating the behavior of several protocols, mostly developed at the routing layer, by using CBR- or FTP-like applications. Just few works (e.g., [5]) try to envision application scenarios in which ad hoc networks could be an enabling factor for applications, rather than a hostile networking environment to cope with.

The whiteboard application exploits services provided by a p2p multicast system. Therefore, we have used a p2p system consisting of an overlay network and a p2p multicast protocol. We have used Pastry to implement the overlay network, and Scribe to build multicast trees on top of the overlay. Pastry and Scribe are among the most efficient solutions developed for the legacy Internet [6]. Finally, we have tested the system both on proactive and on reactive routing protocols (i.e., OLSR and AODV, resp.).

While in previous papers we have focused on the performance of Pastry on ad hoc networks [7–9], in this work we mainly concentrate on the performance of Scribe. Experimental results are discussed in Sections 4 and 5. Specifically, we focus on the QoS provided to WB users in terms of average delay and packet loss. Purposely, we report only results gathered from static ad hoc networks. Taking also into account users mobility would have added further dimensions to the parameters space, making results quite difficult to understand. Our experiments emphasize that proactive routing protocols are much more efficient in our scenario (Section 4). Actually, AODV is practically not able to support any configuration of our testbed. However, even when OLSR is used, Scribe shows severe limitations over ad hoc networks. Even though refined software releases might improve the user QoS (Section 5), there are intrinsic Scribe features that hinder from using it in this networking environment.

We argue that they mainly stem from the design choice of concentrating all the application-level traffic on one single node (the Scribe root node) before delivering it to the final destinations. In fact, from the experimental results, we can conclude that the structured multicast approach used by Scribe is one of the main reasons of its inefficiency. Specifically, structured multicast tends to concentrate the application load on few nodes and links that may become easily overloaded. As a topic of future research, we emphasize that structureless (or stateless) multicast approaches can avoid such concentration, representing a simple and interesting possibility to implement efficient p2p multicast systems in multihop ad hoc networks.

2. RELATED WORK

Experiment-based research on wireless networks, and specifically on multihop ad hoc networks, is gaining momentum in the last few years [10, 11]. Having controllable, reproducible, and reasonable-size wireless testbeds is not trivial. Thus, several research efforts are focusing on how to design and implement testbeds which the whole community can exploit [12–15]. One of the main issues of simulation and theoretical analysis is the accuracy of wireless channel models. Therefore, several papers analyse wireless channel features aiming at providing realistic models (see, e.g., [3] and references herein). Other research efforts target the experimental evaluation of routing [9, 16] or transport [10, 11] protocols on multihop ad hoc networks. Significant effort has also been devoted to build experimental testbeds for mesh networks (see [17] and references herein).

The root of this paper is our previous work on experimental analysis of routing and middleware platforms for multihop ad hoc networks. The work in [7, 8] focuses on issues related to structured overlay networks running on proactive and reactive routing protocols. Specifically, these papers analyse Pastry performance running on top of OLSR [18] and AODV [19]. Work in [7–9] showed that OLSR performs better than AODV in terms of packet loss and delays, when running either a light application such as the ping utility, or a structured p2p system. Furthermore, in [20, 21] we started analysing the performance of a full p2p stack including a complete p2p multicast system based on Scribe and Pastry, and a realistic GC application. In this paper we provide a systematic and comprehensive view of our major findings in using legacy p2p systems to support GC applications in multihop ad hoc networks.

From the study reported in [7–9], the design of an optimised p2p overlay network for multihop ad hoc networks has arisen. We called it CrossROAD [22], since it exploits a cross-layer interaction with a proactive routing protocol. The comparison between Pastry and CrossROAD shows that such interactions can be an enabler to implement structured overlay networks over ad hoc networks in a very efficient way. In this paper we do not focus exclusively on the overlay network, but take into account also the multicast and application layers. To have good hints on how to design an efficient p2p multicast system for multihop ad hoc networks, we do not consider at this stage any possible cross-layer optimisation. Instead, we analyse how a legacy protocol (i.e., Scribe) works in the environment it has been designed for (i.e., Pastry) when used to support GC applications. The results we provide in this paper tell that in our experimental environment, the limits of such a legacy p2p system can be mainly accounted to design choices of Scribe (see Sections 6 and 7). Therefore, simply replacing Pastry with CrossROAD may help because the DHT will become more efficient, but will not solve the problem.

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 (L3) layer, or at the application layer, where application denotes all possible layers above the transport.

In the legacy wired networks, application-level multicast has been introduced to address the practical problems of implementing L3 multicast in the Internet core. L3 multicast requires modification at the routers, since they have to maintain the multicast groups state, which contrasts the original statelessness of the IP protocol. Instead, application-level multicast runs at edge nodes, and just requires standard unicast support from the core.[1] Several application-level multicast protocols have been proposed for the wired Internet (e.g., [23–28]). The most recent proposals among them [24–26, 28] run the multicast protocol on top of an overlay network based on a DHT. This approach is very interesting for

[1] Running exclusively on edge hosts, or end systems, application-level multicast is sometimes referred to as end-system multicast.

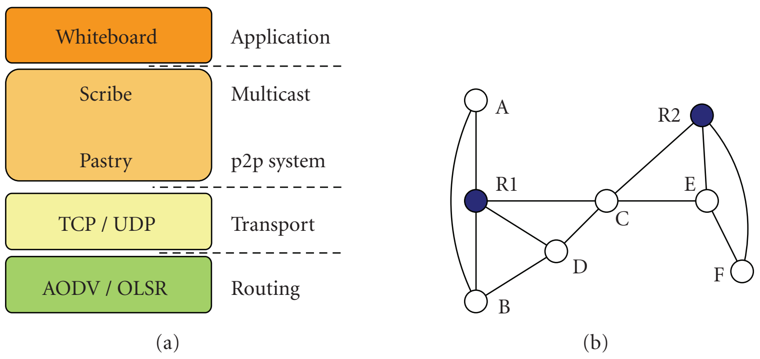


Figure 1: (a) Protocol stack and (b) network topology of our testbed.

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 (as, e.g., in [23]). 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. To the best of our knowledge, the feasibility of such systems on multihop ad hoc networks has not been investigated yet. In this work, we choose Scribe [24], because it is one of the most recent and popular p2p multicast protocols, and has shown to outperform other similar approaches sharing the same concepts [6].

Other papers propose multicast solutions explicitly designed for ad hoc networks (e.g., [29–32], and see the survey in [33]). Also in this case, it is possible to identify L3 approaches (e.g., [31, 32]) and application-layer approaches (e.g., [29, 30]). The argument used to justify application-level multicast in wired networks does not hold in ad hoc networks anymore, as in this case there is no distinction between core and edge hosts. The main reason people highlight to implement multicast at the application level also in ad hoc networks is the fact that in this way, the burden of managing the multicast structure falls exclusively on nodes that are actually interested in being part of the multicast group. However, it should be pointed out that application-level multicast potentially generates path stretch because just a subset of nodes can be used to deliver the data. Moreover, nodes that do not participate in multicast groups have to forward data nevertheless. Therefore, it is not yet clear whether multicast for ad hoc networks should be implemented at the routing or at the application level. To the best of our knowledge, application-level multicast approaches designed for ad hoc networks do not leverage underlying p2p systems as some solutions for wired networks do. We do believe that exploiting DHTs to build multicast systems can be a valid option in ad hoc networks too, due to the advantages highlighted before. This is why we have chosen to evaluate Scribe on ad hoc networks in this paper.

Finally, it is worth mentioning the original branch represented by gossiping protocols applied to multicast (e.g., [34]). The main idea of this approach is that senders of a multicast group select a random subset of nodes in the group to send data to. The same process is repeated at receiving nodes for a given number of turns, this number and the size of the random subsets being protocol parameters. It has been shown that such protocols are actually able to deliver messages with high probability to all intended receivers [34]. A side effect of gossiping is that the message replication rate is quite difficult to control. Evaluating such an approach in comparison with p2p multicasting is an interesting topic which is however out of the scope of this work.

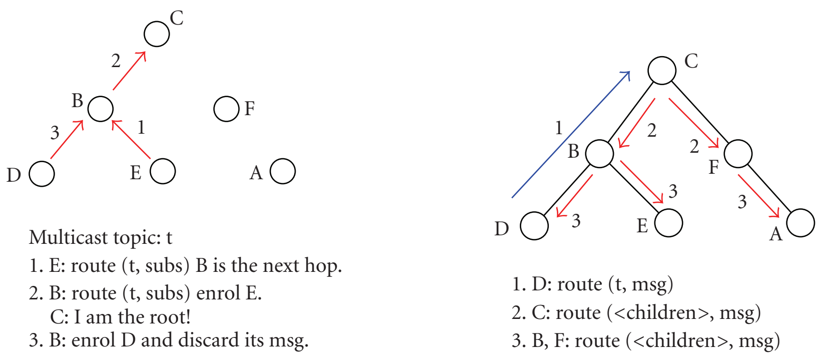
Similarly, in this work we do not specifically consider reliability issues that may arise also in p2p multicasting scenarios (we elaborate this point in Section 3.3). As a future work, it will be interesting to investigate how to efficiently integrate multicast reliability techniques available in literature.

3. EXPERIMENTAL SCENARIO AND SETUP

3.1. Application and protocol stack

One of our targets is to envision realistic applications oriented to multihop ad hoc networks and understand how they could be developed in practice. From this standpoint, GC applications are quite interesting. They fit well the overall features of multihop ad hoc networks since they are distributed, self-organising, and decentralised in nature. As a simple—yet significant—example, we developed a whiteboard application (WB), which implements a distributed whiteboard among the network users. WB usage is very intuitive. Users run a WB instance on their own devices, select a topic they want to join, and start drawing on the canvas. Drawings are distributed to all the devices subscribed to that topic, and rendered on each canvas. WB is just an example of a broader range of applications, including distributed messaging, distributed gaming, and so forth. We believe that this kind of applications can be valuable to users, and they can thus contribute to bring multihop ad hoc networks technologies into everyday life.

WB has been developed on top of the network protocol stack shown in Figure 1(a). Specifically, WB runs on top of a p2p middleware layer made up of a structured overlay network (Pastry [35]), and an application-level multicast


C
2
B
F
3
1
D
E
A
Multicast topic: t
1. E: route (t, subs) B is the next hop.
2. B: route (t, subs) enrol E.
C: I am the root!
3. B: enrol D and discard its msg.
C
1
2
B
2
F
3
3
3
D
E
A
1. D: route (t, msg)
2. C: route (<children>, msg)
3. B, F: route (<children>, msg)

(a) Scribe building the tree.

(b) Scribe delivering data.

Figure 2: Scribe main features.

protocol (Scribe [24]). WB maps each interest group (i.e., each topic) to a multicast tree, and exploits the multicast protocol services to deliver information to group members. Pastry uses both TCP and UDP at the transport layer, and thus we have included both protocols in our architecture. Note that the traffic generated by WB and Scribe is sent over TCP connections. In line with our previous experiments, at the routing layer we used OLSR [18] as a proactive routing protocol, and AODV [19] as a reactive one. The referred implementations of OLSR and AODV provide a standard L3 forwarding platform. Therefore, we could run vanilla TCP and UDP shipped with the Linux kernel on top of them.

Before proceeding, it is useful to give some more details about Pastry and Scribe features. This helps to understand the experimental results presented afterwards.

Pastry implements an overlay network based on a distributed hash table (DHT), on which nodes and data are logically mapped. Specifically, each node gets a logical address within a circular address space as the hashed value of its IP address, while a key is associated to each data to be stored/retrieved in/from the overlay. The onus of defining a key for each message lies with the layer running on top of Pastry (Scribe in this case, or any other distributed application in general). Pastry guarantees that the message is delivered to the node in the overlay with the closest logical address to the hashed key. Thus, the main feature of Pastry is implementing subject-based routing, allowing upper layers to be completely unaware of the eventual-destination address. In more detail, a message generated at a node is sent to the node with the closest address (in the circular space) to the hashed key, to the best of the local-node knowledge. For scalability reasons, in general, each node knows only a subset of the other participants in the same overlay. Thus, the message may reach the final destination following a multihop path on the overlay, possibly resulting in physical-path stretch. Actually, Pastry trades path stretch for scalability and ability of implementing subject-based routing.

In addition, in order to join the overlay, each node has to contact another node already in the overlay, and collect information needed to initialise its internal data structures from the other nodes in the overlay (see [35] for details). In case this procedure fails, the joining node creates its own independent overlay, and has no possibility to rejoin the original network without executing the bootstrap procedure anew. Failures of the bootstrap procedure become quite likely in ad hoc networks, due to the intrinsic instability of wireless links. As shown in the following, we have actually experienced such failures several times.

Scribe has been developed on top of Pastry because the presence of a DHT facilitates the creation and maintenance of the shared trees among groups of nodes. Each tree is identified by a topic. Scribe defines a root node, as the node in the overlay whose address is the closest one to the hashed topic. In Figures 2(a) and 2(b), the root is node C. The tree is built as the union of the reverse paths from the members to the root. Each node willing to join the tree sends a subscribe message specifying the topic as the key. If the local node does not directly know the root of the topic, the message is forwarded using a multihop route at the overlay level. An intermediate node receiving such message either subscribes itself to the same tree by sending its own subscribe message towards the root (e.g., node B after step 1 in Figure 2(a)), or discards the message if it is already a member of the tree (e.g., node B in step 3 in Figure 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 first sent towards the root of the topic (step 1 in Figure 2(b)), and are subsequently delivered by each parent to its children (steps 2 and 3 in Figure 2(b)). Parent-child relations are periodically refreshed by parents sending heartbeats messages to children. Application messages are also used as implicit heartbeats. A child missing heartbeats or application messages for a given timeout value assumes that the parent has left the network, and subscribes again.

As a final remark, we have used the Pastry and Scribe implementations provided by the Rice University in the FreePastry package [36].

3.2. *Network topology*

Figure 1(b) shows the network topology we used. In each trial, nodes A to E ran the whole protocol stack, including the WB application, while nodes R1 and R2 just worked as routers. We think this is a reasonable scenario for standard standalone multihop ad hoc networks. Specifically, it lies within the ad hoc horizon defined in [4], that is, up to 10–20 nodes, and up to 2-3 hops. Theoretical [37] and experimental [4] results show that flat multihop networks beyond this horizon are unable to deliver reasonable throughput to users, and thus they are not likely to be really deployed.

Note that in a small-scale network as the one we consider, each node becomes aware (at the Pastry level) of all the other nodes. This is mainly because Pastry is designed for very large-scale networks, and the subset of the other nodes locally known by each node is typically 32. Thus, in our testbed all the nodes ended up being just one hop away from each other in the overlay network.

Even though this testbed represents a small-scale network, it already highlights main limitations of legacy multicast p2p solutions on multihop ad hoc networks. We can thus expect these problems to significantly exacerbate in large-scale networks, or when multiple multicast groups are concurrently active in the network, or in case of mobility. The main point we try to make in this paper is the fact that even in small-scale realistic multihop ad hoc networks, legacy p2p multicast systems are able to deliver sufficient QoS to user applications just at light traffic loads.

3.3. *Experiments definition and performance indices*

In our experiments all the nodes were IBM ThinkPad R50 laptops with an integrated 802.11b wireless card (Intel PRO-Wireless 2200). The OS was linux-2.6.12.3, loading the ipw2200 driver for the network card. The experiment software can be downloaded from http://bruno1.iit.cnr.it/scribe_exp_sw/.

In all the experiments, nodes A to E ran the WB application while the others just worked as routers. Specifically, the "WB nodes" tried to join a single overlay and consequently a single tree related to a specific topic of interest. In our configuration, every node always assumed the same logical identifier obtained by hashing its IP address, and the topic used by the WB users was always the same.

Under the hypothesis that Pastry generated a single overlay encompassing all WB nodes, the root of the Scribe tree (i.e., the node whose ID is closest to the WB topic ID) was the same through all the experiments, and was node C in Figure 1(b). Users interactions with the WB were simulated by a software agent that alternated between active and idle phases. Specifically, in each active phase the agent generated traffic on the network corresponding to strokes drawn on the WB. Both the number of strokes drawn during an active phase, and the duration of an idle phase were exponentially distributed. Such a traffic profile is bursty, representing the typical behavior of a user that sends content to be shared with the group, and then "idles," looking at data generated by other users.

We ran experiments by varying both the average idle-phase duration, and the average number of strokes per active phase. Each trial was composed by 100 active/idle cycles, and we took care that each node running WB generated at least 100 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. Then, the Pastry bootstrap sequence occurred as follows: node C started first, and generated the overlay. Nodes E and D started 5 seconds after C, and bootstrapped from C. Node B started 5 seconds after D and bootstrapped from D. Node A started 5 seconds after B and bootstrapped from B. Finally, node F started 5 seconds after E and bootstrapped from E. After all nodes joined the overlay, the Scribe tree was created and, finally, WB instances started sending application messages.

The time intervals used for the bootstrap sequence were defined to reduce the probability of failures during the bootstrap procedure. Furthermore, the bootstrap sequence was defined to make each node bootstrap from a physical neighbour. This is a quite realistic assumption, and also increases the probability of nodes to correctly complete the bootstrap procedure. However, often this was not sufficient. In fact, the instability of the wireless links caused high packet loss resulting in TCP connection failures and the consequent generation of an isolated overlay.

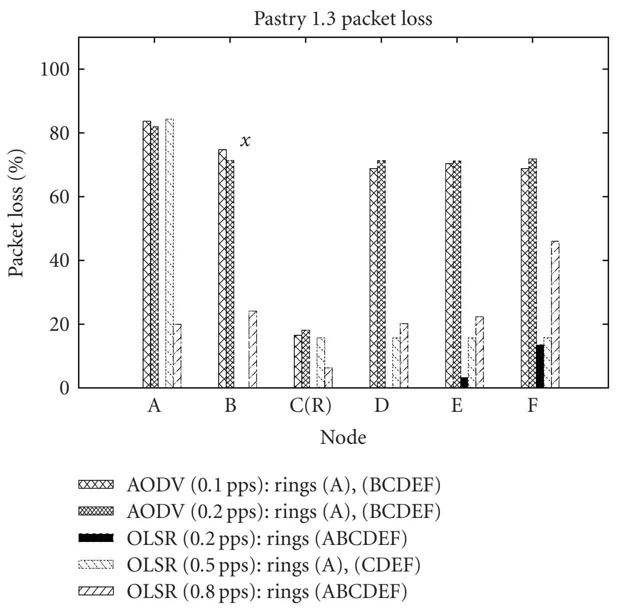
In the performance analysis, each trial is identified by the routing protocol used and by an application-load index, measured as the number of packets per second (pps) generated by each user. This index is defined as the ratio between the average number of strokes generated in the active/idle cycle, and the average duration of the cycle. We have found that this simple index is sufficient to correctly identify usage cases of a GC application in our scenario.

The main performance indices presented in the following are the packet loss and the average delay experienced by each node during the experiment. The packet loss is defined as $1-(R_i/\sum_{j=1}^{N} S_j)$, where R_i is the number of messages received by node i, and S_j is the number of messages transmitted by the jth node. Since Pastry uses TCP at the transport layer, one could expect not to see any packet loss. Actually, Pastry uses an internal queue to store messages going to be sent. Packet loss actually occurs when this queue fills up, and is thus a side effect of network congestion.[3] The average delay experienced by node i is defined as $\sum_{j=1}^{R_i} d_{ij}/R_i$, where d_{ij} is the delay experienced by node i in receiving packet j.

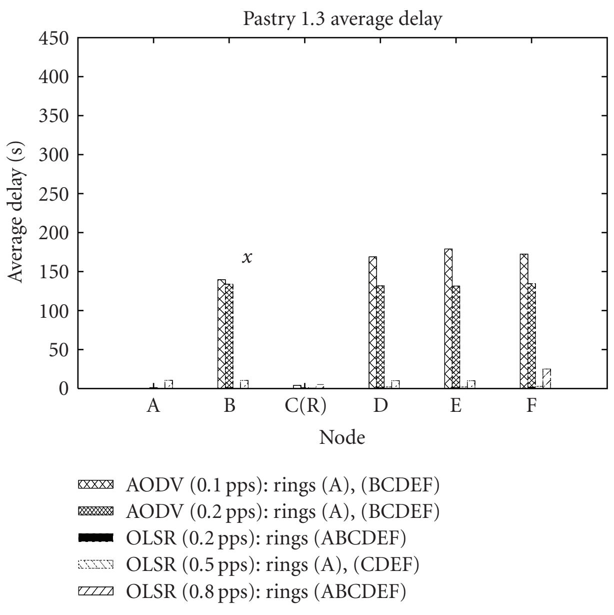
We have also defined a usability threshold for the application, indicating reference values for both delays and packet loss. Beyond these thresholds, the application performance is deemed not compliant with users expectations. To have reasonable values, in our case we assume 10-second delay

[2] A distinct message was sent for each stroke. The size of each message was 1448 bytes.

[3] Properly dimensioning the queue to find the right balance between delay and packet loss depends on the particular application demands (actually, in other set of experiments, we have completely removed packet losses by allowing the queue to grow unlimited).



(a) Packet loss.



(b) Average delay.

FIGURE 3: Pastry performance on AODV and OLSR.

and 15% packet loss as thresholds. Of course they closely depend on the specific application requirements. We replicated each configuration several times, obtaining quite variable results. They are mainly due to the variability of the wireless medium. In this paper, we show the best results measured in each configuration.

4. IMPACT OF ROUTING PROTOCOLS ON SYSTEM PERFORMANCE

Results presented in this section aim at evaluating the impact of the underlying routing protocol on the multicast tree creation and maintenance. Figures 3(a) and 3(b) show the packet loss and the delay indices experienced by the WB nodes at different traffic loads. Specifically, we consider AODV experiments with nodes generating 0.1 pps and 0.2 pps, and OLSR experiments with nodes generating 0.2, 0.5, and 0.8 pps, respectively. System performance running AODV is quite bad even with such a light traffic load, thus there was no reason to further increase it. An "*x*" label for a particular node and a particular experiment denotes that for that experiment, we are not able to derive the index related to the node (e.g., because some component of the stack crashed during the experiment).

First of all, the impact of the Pastry bootstrap procedure should be highlighted. In both cases, we experienced several failures during the bootstrap procedure of Pastry that highly influence the system performance. The bootstrap procedure is actually a critical point for Pastry on wireless networks. In this phase, the bootstrapping node has to initialise its internal data structures by opening TCP connections with several other nodes, and gathering portions of their internal data structures. The intrinsic instability of wireless links results in possible failures of some of these connections, which prevents the bootstrap procedure to successfully complete, and the node to join the overlay. Such events were quite frequent in our testbed. When a node fails to join the overlay, it also creates an independent tree and it is not able to receive WB messages from the other participants in the main overlay. This generates a high packet loss on the isolated node, and increases the packet loss also on the other nodes that cannot receive the messages of the isolated node. For this reason, we report in the legend of the plots the overlays configuration built during each trial. Each experiment related to a specific traffic load has been repeated several times observing different outcomes of the bootstrap procedure. We report in this paper the results obtained selecting the experiments in which the number of overlay partitions is minimised. This allows us to focus the performance evaluation mainly on the multicast protocol and the application usability, limiting the effects of problems during the Pastry bootstrap phase. From the experiments shown in Figure 3, we can note that the probability of bootstrap failure is higher running AODV than OLSR, even at lighter traffic loads. This is due to high delays required by AODV to establish a new route towards a destination, especially with unstable links, causing TCP-connection failures [9, 38].

Failures of the bootstrap procedure have a significant effect on packet loss (Figure 3(a)). In fact, in both AODV experiments, node A is isolated and creates its own overlay. This results in packet loss higher than 80% at node A (i.e., it just receives its own WB messages, which counts for about one sixth of the overall WB traffic). The packet loss is about 16% at node C (the root node), and about 70% at the other nodes. Qualitatively, similar remarks apply to the "OLSR 0.5 pps" experiment mainly because some software component crashed on node B during the experiment, and node A was forced to fail its join operation trying to connect to node B. Otherwise, in the other experiments running OLSR, all nodes correctly joined the overlay, giving a complete view of WB performance. Specifically, in the case "OLSR 0.2 pps," the packet loss is 0 at nodes A, B, C, and D, while it is about 3% at node E, and about 13% at node F, since its connection with the root node is quite less stable than the other nodes' connection with the root node. In the case "OLSR 0.8 pps," the packet loss is higher at all nodes. Specifically, node C measures a packet loss of about 6%, nodes A, B, D, and E measure about 20% packet loss, and node F about 45% packet loss. The increased packet loss stems from an architectural design choice of Scribe. Due to the Scribe algorithm, each WB message to be distributed on the tree is firstly sent to root, and then forwarded over the tree. Often, this is an excessive load for the root node, which, as the application load increases, becomes unable to deliver all the received messages, and drops them at the sending queue. This event not only depends on the traffic load generated by the application, but also on the routing protocol used. The fact that the root node drops messages at the *sending* queue is also the reason why the root node always experiences lower packet loss with respect to the other nodes.

Even though we have replicated the experiments in each configuration several times, we have not been able to make all the nodes execute the bootstrap procedure correctly in the case "OLSR 0.5 pps," while we have been able in the case "OLSR 0.8 pps." This actually *does not* mean that the system works for a higher load (0.8 pps), and does not for a lower load (0.5 pps). In fact, the bootstrap procedure is not influenced by the application load, because it is executed before the application starts. Therefore, the probability that all nodes correctly bootstrap exclusively depends on the links' stability. From an usability standpoint, results shown for the case "OLSR 0.5 pps" are useful even if the network configuration is different from the other OLSR cases. The packet loss measured on the main overlay is essentially due to the isolation of node A, while the other nodes are able to receive almost all messages from each other. On the other hand, the packet loss becomes clearly too high at 0.8 pps. We can thus conclude that the system is surely not usable beyond 0.5 pps in case of OLSR, while the threshold clearly drops to 0.1 pps in case of AODV. In this case, besides never being able to correctly bootstrap, the system drops many messages also in the main overlay network. Thus, as far as the packet loss is concerned, we can conclude that OLSR outperforms AODV, because it makes the overlay more stable. It should be pointed out that the better performance of OLSR with respect to with AODV confirms our previous findings, even in mobile configurations (see [7] as an example). This is mainly because AODV builds routes trying to use also unidirectional and asymmetric links. Therefore, in our experiments the network turned out to be always far less stable with AODV than with OLSR.

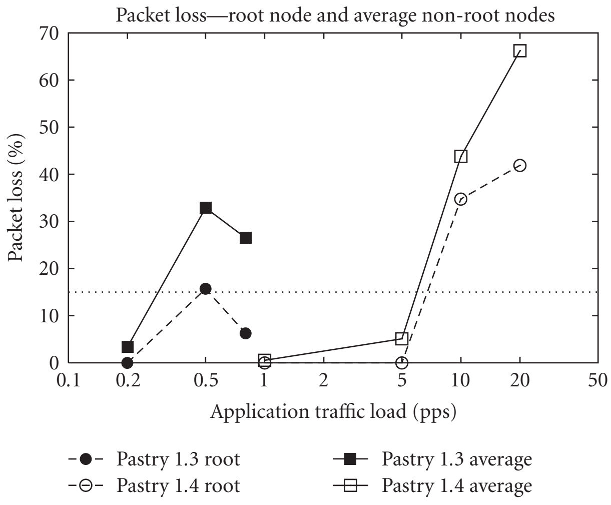
Similar observations can be drawn by focusing on the delay index (Figure 3(b)). First of all, it should be pointed out that the delay related to nodes that are the sole member of their own overlay (e.g., node A in the "AODV 0.1 and 0.2 pps" case) is obviously negligible. Furthermore, it is confirmed that OLSR performs better than AODV. In fact, while AODV leads to average delays of the order of 100 seconds, OLSR produces delays of at most 20 seconds, at 0.8 pps. Again, the root node (C) always experiences a lower delay with respect to the other nodes in the same overlay. From OLSR experiments, we can note that at 0.5 pps, the maximum delay is about 2 seconds, while at 0.8 pps, node C measures an average delay of about 5 seconds, nodes A, B, D, and E measure about 10 seconds, and node F about 25 seconds. This confirms the packet loss analysis, also with respect to the usability thresholds.

All these experiments have been run using the FreePastry 1.3 release of Pastry code. Even though the experiments have been run in a particular setup, and the results refer to particular experiments, the outcome of our measurements can be generalised fairly well. Actually, we can conclude that the Pastry/Scribe platform is practically unable to support even light application loads, even though the system performance running OLSR demonstrates that a proactive approach at the routing layer can increase the usability threshold. However, in the last few months, new versions of FreePastry have been released, and we decided to replicate this analysis using one of the latest versions (1.4.1) to highlight possible improvements introduced by software refinements of Pastry. In the next section, we present a comparison of the two releases. As OLSR drastically outperformed AODV in all cases, we used only OLSR in this new set of experiments.

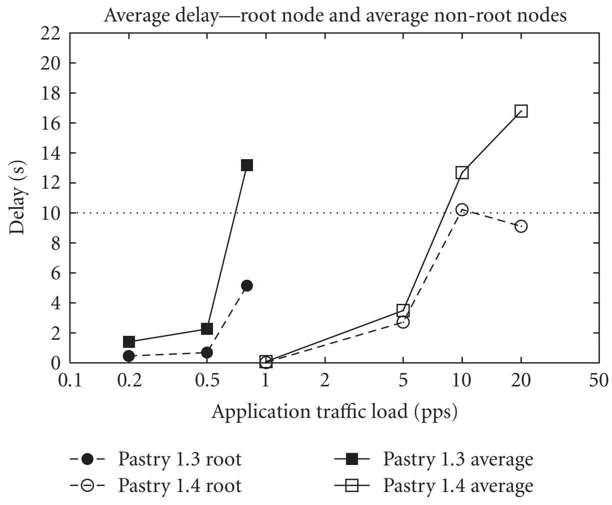
5. IMPACT OF PASTRY SOFTWARE REFINEMENTS

In order to compare system performance depending on the software release, we do not analyse packet loss and delays per node, but we consider the same indices for the root node, and an average value for all the other nodes. The root node represents the best case of the system performance since each message has to be firstly sent to it and then forwarded to the others.

Figures 4(a) and 4(b) show the performance we measured in terms of average delay and packet loss, respectively, for both software releases. Focusing on "Pastry 1.3" curves, we can summarise the analysis presented in the previous section. The root node experiences a reasonable QoS for all application loads (i.e., 0.2 pps, 0.5 pps, and 0.8 pps). Specifically, the root-node average delay is always below 5 seconds, and the packet loss below 15%. But other nodes obtain a largely unsatisfactory service. We can identify a critical point for an application load between 0.5 and 0.8 pps. At 0.5 pps, the system performance is highly influenced by the bootstrap failure of node A. As far as the packet loss is concerned, isolation of node A raises the average packet loss of non-root nodes to 32.86%. However, the average delay experienced by node A is clearly negligible. Thus, the delay averaged over non-root nodes, which is about 2.3 seconds, is a lower bound of the real delay that non-root nodes would have experienced under this traffic load if the overlay network had been built correctly. When the application load increases to 0.8 pps, all nodes belong to the same overlay. The packet loss slightly



(a) Packet loss.



(b) Average delay.

Figure 4: Pastry releases comparison.

decreases to 26.50% at the expenses of a sharp increase of the average delay, which raises to 13.17 seconds. Hence, we can confirm that under Pastry 1.3, the system is reasonably usable only for very light application loads, and deploying applications like WB on this platform becomes quite questionable.

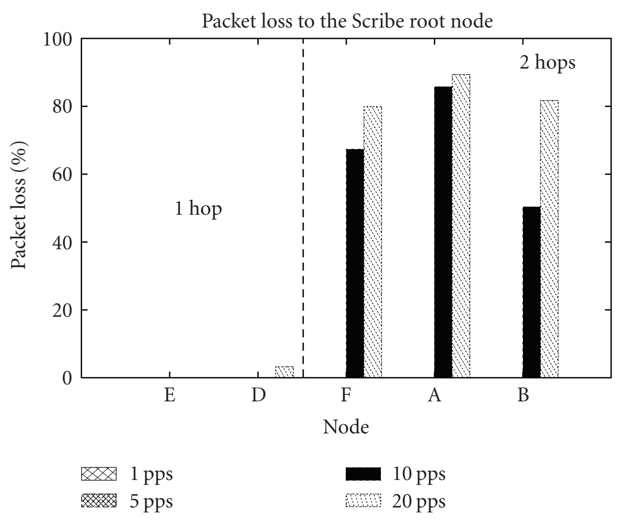
In order to further reduce the impact of Pastry inefficiencies on the system performance, in the second set of experiments running the new software release we discarded the trials affected by bootstrap failures, only considering the formation of a single overlay, even though this represents an optimistic assumption in real experiments. This might sound unfair to the FreePastry 1.3 version. However, it should be pointed out that even considering only experiments in which the bootstrap procedure correctly completed, FreePastry 1.3 is clearly unusable at 0.8 pps, while FreePastry 1.4 remains usable for much greater application loads, as shown by Figures 4(a) and 4(b).

Analysing "Pastry 1.4" results, we noticed that the major modifications to the overlay building and maintenance procedures have drastically reduced the overhead and improved the overlay stability [36]. Thus, it has been interesting to explore whether this new release improves also the application performance in our scenario.

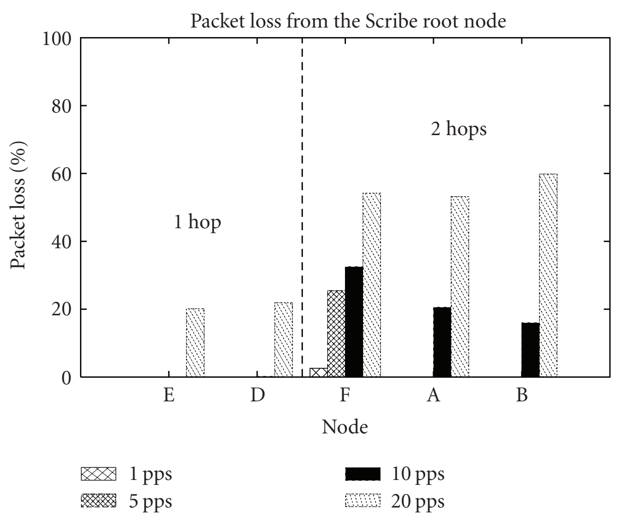
By looking at Figures 4(a) and 4(b), the performance improvement is evident. The critical point moves by about one order of magnitude, lying now between 5 and 10 pps. Indeed, at 5 pps also non-root nodes experience reasonable QoS, since the average delay is about 3.5 seconds, and the packet loss is 5%. On the other hand, at 10 pps and beyond the application becomes hardly usable at any node. Note that even though the average delay at root node would be almost acceptable also at 10 and 20 pps (i.e., it is below the usability threshold), the packet loss increases to 35% and 42%, respectively.

Note also that between 10 pps and 20 pps, the delay curve relative to the root node flattens. This is actually a side effect of the higher packet loss experienced at 20 pps. In detail, in the topology of our experiments, non-root nodes are either 1 or 2 hops away from root. In the next section, we show that even at 20 pps, 1-hop-away nodes are able to send to root almost all the traffic locally generated. The additional packet loss experienced by root at 20 pps is thus due to less messages received from 2-hop-away nodes. In other words, out of the whole bunch of messages received by root, the fraction of messages received by 1-hop-away nodes *increases* when the traffic load shifts from 10 to 20 pps. Since messages from 1-hop-away nodes experience significant lower delay than messages from 2-hop-away nodes, the fraction of messages experiencing low delays increases too. This reduces the average delay measured at root, resulting in the flat shape of the curve. The same phenomenon does not apply to non-root nodes. Recall that messages have to reach the root before being delivered to the other nodes. The delay experienced by non-root nodes in receiving messages *from root* increases between 10 and 20 pps. Thus, even though in both cases messages experience—on average—the same delay between the originating node and root, in the 20 pps case they undergo higher delay along the path between root and the final destination.

To summarise, the above analysis shows a steep improvement in the user-perceived QoS when moving from a FreePastry release to a more advanced one. However, a critical point still exists, beyond which it practically makes no sense to use GC applications like WB on this platform. At this point, it is really important to understand whether this critical point is going to eventually disappear thanks to future software releases, or it is intrinsic in the Pastry/Scribe design. In the following sections, we analyse the results achieved with FreePastry 1.4.1 more in depth, and show that independently of software refinements, the design of Scribe includes features not suitable for the multihop ad hoc networks environment.



(a) Packet loss towards root.



(b) Packet loss from root.

Figure 5: Packet loss analysis.

6. ROOT AS THE CENTRAL NODE: A RATHER OPTIMISTIC SETUP

In the set of experiments presented so far (related to Pastry 1.4.1), we have placed the root node at the center of the topology to minimise the average hop distance to any other node. Since it is well known that TCP performance drastically worsens as the hop distance increases [4, 38], this represents an optimistic setup. To have a clearer picture of the system behavior, we now focus on the average delay and packet loss experienced by each single node. Specifically, curves in Figures 3(a) and 3(b) show the average performance experienced by non-root nodes, and thus provide indications about the average QoS a user may expect. In this section we analyse the performance of nodes at 1 and 2 hops from root separately. Together with the "root-curves" in Figures 3(a) and 3(b), this provides a precise view of the expected QoS with respect to the position of a node in our topology.

Figures 5(a) and 5(b) show the packet loss experienced by each single node *towards* and *from* root, respectively. The packet loss of the *i*th node *towards* root is defined as the ratio between the number of messages generated at the *i*th node and not received by root, and the number of messages generated at the *i*th node, that is, $1-(R_r^{(i)}/G^{(i)})$, where $R_r^{(i)}$ is the number of messages generated by the *i*th node and received by root, and $G^{(i)}$ is the number of messages generated by the *i*th node. The packet loss of the *i*th node *from* root is defined as the ratio between the number of messages not received by the *i*th node out of the total number of messages sent by root, and the total number of messages sent by root, that is, $1-(R_i^{(r)}/S^{(r)})$, where $R_i^{(r)}$ is the number of messages sent by root and received by the *i*th node, and $S^{(r)}$ is the number of messages sent by root. Note that due to the Scribe architecture, root not only sends messages locally generated, but also messages that it receives from all the other nodes. Due to packet loss towards root, the number of messages sent by root is less than the sum of messages generated by the sender nodes, that is, $S^{(r)} < \sum_{i=1}^{N} G^{(i)}$. In addition, the *i*th node experiences packet loss 0 only if (i) the packet loss of *all* nodes *towards* root is 0, and (ii) the packet loss of the *i*th node *from* root is 0.

On the one hand, Figures 5(a) and 5(b) confirm that as far as the packet loss is concerned, the critical point for WB usability lies between 5 and 10 pps. Indeed, below 5 pps the packet loss *towards* root is 0 at all nodes. This means that all nodes are able to send messages locally generated to root. Apart from node F, also the packet loss *from* root is 0 at all nodes. Thus, the overall packet loss experienced by all nodes except F is 0 at 5 pps and below, making WB usable.

On the other hand, Figures 5(a) and 5(b) show a drastic difference between nodes at 1-hop and 2-hop distance from root. Even though the magnitude of this difference is quite surprising, a steep performance decrease in the case of multi-hop connections was expected (see, e.g., [4, 38]). Specifically, Figure 5(a) shows that even at 10 pps and 20 pps, 1-hop-away nodes are able to send almost all their messages to root. Instead, 2-hop-away nodes see their outgoing traffic cut by 50% to 89%. Figure 5(b) shows a similar trend, in the sense that at 10 and 20 pps, 2-hop-away nodes experience a far higher packet loss than 1-hop-away nodes. However, there is a difference worth to be noted. While for 2-hop-away nodes the packet loss is higher in the direction *towards* root, for 1-hop-away nodes the packet loss is higher in the direction *from* root. At a higher level, one could note that as the traffic load increases, Scribe cuts it because the root node becomes overloaded. In our configuration, while 1-hop-away nodes suffer only in the direction *from* root, 2-hop-away nodes mainly suffer in the direction *towards* root. Actually, we have found configurations in which for both cases (i.e., 1 and 2 hops), the main traffic cut occurs in the direction *from* root. Understanding the reason of this behavior is not trivial, thus we are currently analysing the system even more deeply. However,

Table 1: Delays depending on the hop distance from root (seconds).

	Average/percentiles	Root	1 hop	2 hops
1 pps	Average	0.032	0.055	0.100
	90	0.070	0.115	0.221
	95	0.126	0.159	0.316
	99	0.282	0.321	0.604
5 pps	Average	2.710	2.900	3.924
	90	11.00	11.29	13.15
	95	13.74	13.95	16.62
	99	23.19	23.28	25.71
10 pps	Average	10.22	10.57	14.52
	90	34.06	34.79	43.11
	95	65.06	65.11	71.36
	99	101.4	101.4	101.8
20 pps	Average	9.11	13.12	21.15
	90	23.09	31.11	88.03
	95	86.30	89.30	115.1
	99	145.4	145.7	146.4

it should be noted that as far as the application-level QoS is concerned, the precise direction along which the main traffic cut occurs is not that important.

Figures 5(a) and 5(b) finally show that the presence of 2-hop-away nodes makes the application unusable also for 1-hop-away nodes. For example, let us focus on the 10 pps case. Nodes 1 hop away from root measure 0 packet loss on both directions. However, they are unable to receive most of the messages generated at nodes 2 hops away from root, because those nodes suffer very high packet loss *towards* root. This highlights that since all messages have to be firstly sent to the root, a poor connection between a particular node and root makes *all* other nodes unable to receive the messages generated by this node.

Analysing the delay figures allows us to highlight a further feature of the system. Specifically, Table 1 shows the average delay and the main percentiles depending on the application-traffic load, and on the hop-distance from root. By looking at the average delays only, one could conclude that the application is usable even at 10 pps by nodes at most 1 hop away from root. However, if the usability threshold is defined with respect to the 90th percentile instead of the average value, the critical point shifts below 5 pps (for all nodes).

Table 1 also shows a drastic difference between 1 pps and the other traffic loads. Indeed, at 1 pps WB performance is completely satisfactory, as the 99th percentile for 2-hop-away nodes is about 600 milliseconds. At higher traffic loads, there is a significant difference between the average delays and the 90th percentiles. This suggests that the delay distributions have a long tail. This is confirmed by looking at Figure 6(a), which plots the CCDF of delays measured at root node. Clearly, for each traffic load, the CCDF at root is a lower bound of CCDFs at any other node. Figure 6 shows that CCDFs for application loads of 5 pps and beyond can be lower bounded by a Pareto distribution with parameter 0.25. Specifically, they show a long-tail pattern in the range from 5 milliseconds to 10 seconds. Figures 6(b) and 6(c) show the same feature for 1-hop-away and 2-hop-away nodes, as well. Being the coefficients of the approximating Pareto distribution far below 1, the tail is very heavy also in these cases. At the application level, this means that even though the average delay can lay below the usability threshold, delay values are highly variable, and thus no strict QoS guarantees can be granted.

From results presented in this section, we can conclude that the centralised approach of Scribe generally hinders the use of GC applications over multihop ad hoc networks. Specifically, we have highlighted two main inefficiencies.

(i) Even few nodes poorly connected to root prevent *all* the nodes from using GC application properly. This is because those nodes will experience very high packet losses towards the root, and all the other nodes will be unable to receive their messages. Even nodes that could be physically very close to the "poorly connected" nodes, and thus might be potentially able to correctly communicate with them, will just get a small fraction of their messages.

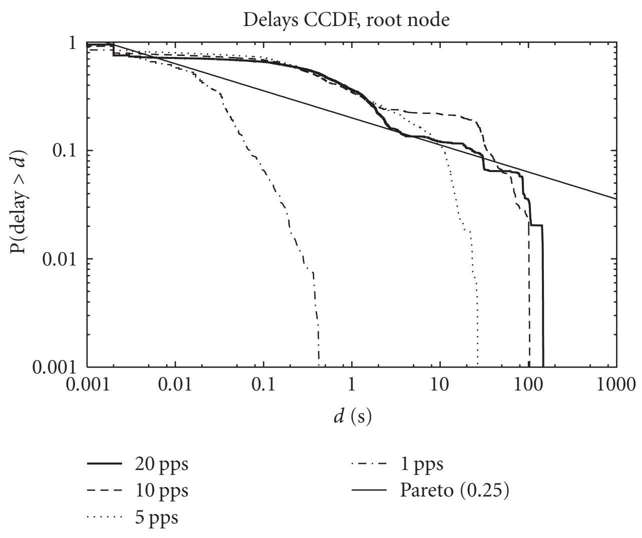
(ii) The root node very likely experiences a long-tailed delay distribution. In these cases, *any* other node experiences the same pattern in the delay distribution.

Both of these drawbacks are intrinsic to the Scribe design, and cannot be completely addressed either by refined software releases, or by simply improving the underlying DHT. Note also that our experiments were run with quite powerful laptops. Performance could thus be even worse in case of less powerful devices, such as PDAs, that are natural candidates to be used in multihop ad hoc networks environments.

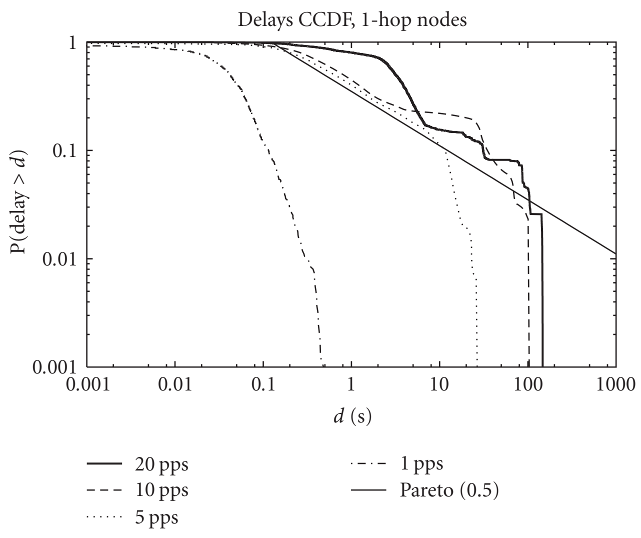
7. SYSTEM PERFORMANCE VARYING THE SCRIBE ROOT NODE LOCATION

In the experiments presented so far, the Scribe root node was always at the center of the network topology and all the other nodes were 2 hops away at most. However, in the general case no assumptions about the root location can be done. Therefore, we ran a further set of experiments by placing the root node at one edge of the network. This might sound like a pessimistic configuration, but it should be noted that having the root node at one edge of the network is more likely than having it at the center of the topology. In detail, with respect to the topology in Figure 1(b), we swapped the positions of nodes C (root node) and A. This setup also allows us to better analyse the impact of longer paths on Scribe, since we have TCP connections spanning 1 to 4 hops.

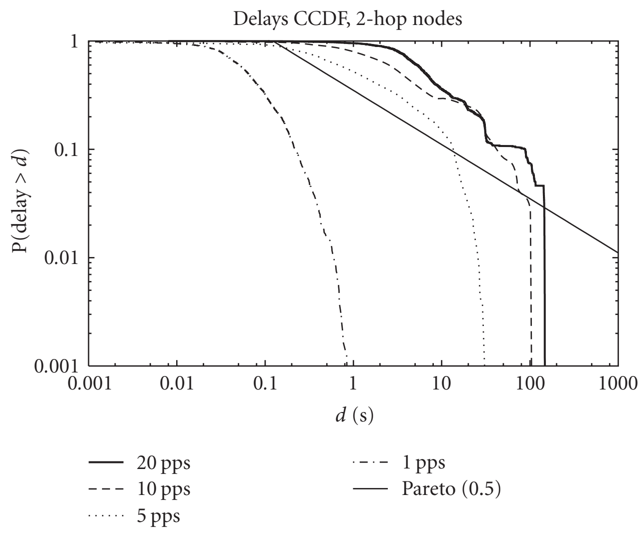
Figures 7(a) and 7(b) show the performance measured at each single node in terms of average delay and packet loss, respectively. They confirm that the system now is usable just at lighter application loads. Even at 1 pps, while the packet loss is 0 at all nodes, the average delay ranges between 5 and 10 seconds. Thus, even at 1 pps, the system is usable just for applications with loose delay constraints. At a load of 5 pps, the system is completely unusable for nodes 3 hops away from root and beyond. At higher loads, the performance might be too low even for the root node.



(a) Root node.



(b) 1-hop-away nodes.



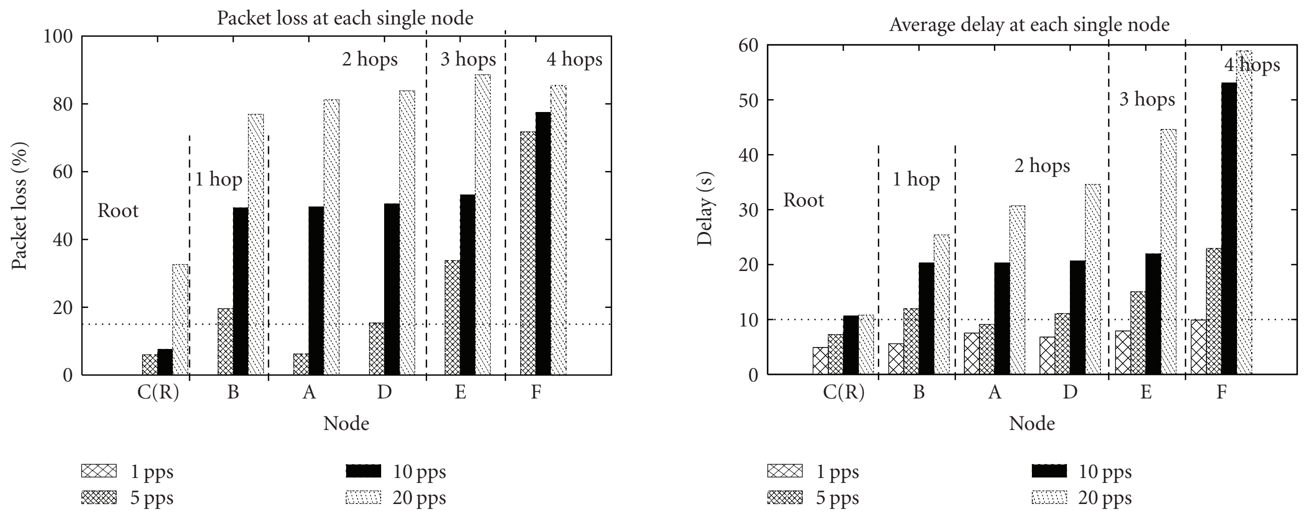
(c) 2-hop-away nodes.

FIGURE 6: Delay CCDFs.

8. DISCUSSION AND FUTURE WORK

Scribe was designed having in mind a resource-rich environment, such as the legacy wired Internet, and it has shown to perform very well in it [6]. However, when used over multihop ad hoc networks, the centralised approach, in which a root node is in charge of delivering all application messages, leads to very low performance in terms of packet loss and delay. One might think of using solutions like SplitStream [28], which splits a single multicast group in several trees, thus reducing the concentration at the root. However, SplitStream requires significant a priori knowledge about the application traffic load, and a quite significant planning effort. Actually, it is again a system designed for large-scale wired networks, which introduces even further networking structures to the shared tree used by Scribe.

In general, we believe that one of the main reasons of the low performance is indeed the use of a *structured* multicast solution. Besides requiring significant overhead in terms of management traffic, such solutions tend to concentrate the costs of the application (in terms of network/computation resources) on a few nodes and links. If these nodes/links are underprovisioned, or happen to be placed in adverse locations, the whole system may implode, making it unable to support the application. Structured solutions are a good choice in large-scale systems designed for the legacy Internet (as Scribe is). Indeed, in this case the network and computational resources are not a big issue, and structured solutions allow the system to scale up to thousands of nodes. However, growing up to such a scale is not reasonable for multihop ad hoc networks. Theoretical results and practical experiences (e.g., [4, 37]) show that the most reasonable scale for



(a) Packet loss (b) Average delay

Figure 7: Single-node analysis.

flat ad hoc networks based on 802.11 technologies is up to 10–20 nodes, and 2-3 hops. Furthermore, concentrating the load on a few resources and managing the multicast structure may become a serious problem in these networking environments.

Based on these remarks, we believe that *structureless* (or *stateless*) multicast is a more reasonable choice in this environment. For example, approaches like differential destination multicast (DDM, [39]) and route-driven gossip (RDG, [34]) seem to be very interesting solutions. Typically, according to this approach, each member of a multicast group knows the other members of the same group (actually, RDG also works with partial knowledge). When a message is locally generated, it is sent to the group members by using the underlying routing protocol, without requiring any multicast structure. Mechanisms are included to send just once a message addressed to destinations sharing an initial portion of the path from a sending node. These approaches spread the load more evenly over the nodes and links of the network, avoid concentration on a few nodes/links, and typically work remarkably well in small- and medium-size networks. However, they require costly mechanisms to collect and maintain the (partial) list of group members at each node. Typically, nodes have to periodically flood the network to announce their presence, or to check for other nodes' liveness. Therefore, we are currently designing an improved version of Scribe, which retains the structureless features of DDM and RDG, but avoids such costly mechanisms by exploiting a cross-layer approach. In this view, further advantages would be brought in by replacing Pastry with CrossROAD. In all previous experiments, CrossROAD has shown to outperform Pastry in multihop ad hoc networks by fixing all its inefficiencies (e.g., path stretches, bootstrap failures, nodes' isolation, etc.). Since the results presented in this paper clearly show that even one of the best legacy p2p multicast system needs drastic improvements to work in ad hoc networks, using CrossROAD to support an enhanced p2p multicast system is a natural choice.

REFERENCES

[1] M. Gerla, C. Lindemann, and A. Rowstron, "P2P MANET's - new research issues," in *Perspectives Workshop: Peer-to-Peer Mobile Ad Hoc Networks - New Research Issues*, M. Gerla, C. Lindemann, and A. Rowstron, Eds., Dagstuhl, Germany, April 2005.

[2] *ACM Mobicom MobiShare Workshop*, September 2006, http://www.mobishare.org/.

[3] G. Anastasi, E. Borgia, M. Conti, E. Gregori, and A. Passarella, "Understanding the real behavior of Mote and 802.11 ad hoc networks: an experimental approach," *Pervasive and Mobile Computing*, vol. 1, no. 2, pp. 237–256, 2005.

[4] C. Tschudin, P. Gunningberg, H. Lundgren, and E. Nordström, "Lessons from experimental MANET research," *Ad Hoc Networks*, vol. 3, no. 2, pp. 221–233, 2005.

[5] E. Huang, W. Hu, J. Crowcroft, and I. Wassell, "Towards commercial mobile ad hoc network applications: a radio dispatch system," in *Proceedings of the 6th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*, pp. 355–365, Urbana-Champaign, Ill, USA, May 2005.

[6] M. Castro, M. B. Jones, A.-M. Kermarrec, et al., "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 2, pp. 1510–1520, San Francisco, Calif, USA, March-April 2003.

[7] E. Borgia, M. Conti, F. Delmastro, and E. Gregori, "Experimental comparison of routing and middleware solutions for mobile ad hoc networks: legacy vs cross-layer approach," in *Proceeding of the ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis*, pp. 82–87, Philadelphia, Pa, USA, August 2005.

[8] E. Borgia, M. Conti, F. Delmastro, and A. Passarella, "MANET perspective: current and forthcoming perspective," in *Proceedings of the 15th IST Mobile & Wireless Communications Summit*, Mykonos, Greece, June 2006.

[9] E. Borgia, "Experimental evaluation of ad hoc routing protocols," in *Proceedings of the 3rd IEEE International Conference*

on Pervasive Computing and Communications Workshops (PerCom '05), vol. 2005, pp. 232–236, Kauai Island, Hawaii, USA, March 2005.

[10] *IEEE ICPS REALMAN Workshop*, Santorini, Greece, July 2005, http://www.cl.cam.ac.uk/realman/05/.

[11] *Proceeding of the ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis*, Philadelphia, Pa, USA, August 2005, http://www-ece.rice.edu/E-WIND/.

[12] "APE: Ad Hoc Protocol Evaluation Testbed," Department of Computer Systems at Uppsala (Sweden). http://apetestbed.sourceforge.net/.

[13] N. H. Vaidya, J. Bernhard, V. V. Veeravalli, P. R. Kumar, and R. K. Iyer, "Illinois wireless wind tunnel: a testbed for experimental evaluation of wireless networks," in *Proceeding of the ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis*, pp. 64–69, Philadelphia, Pa, USA, August 2005.

[14] K. Ramachandran, S. Kaul, S. Mathur, M. Gruteser, and I. Seskar, "Towards large-scale mobile network emulation through spatial switching on a wireless grid," in *Proceeding of the ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis*, pp. 46–51, Philadelphia, Pa, USA, August 2005.

[15] P. De, A. Raniwala, S. Sharma, and T.-C. Chiueh, "MiNT: a miniaturized network testbed for mobile wireless research," in *Proceedings of the 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 4, pp. 2731–2742, Miami, Fla, USA, March 2005.

[16] R. S. Gray, D. Kotz, C. Newport, et al., "Outdoor experimental comparison of four ad hoc routing algorithms," in *Proceedings of the 7th ACM Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '04)*, pp. 220–229, Venezia, Italy, October 2004.

[17] R. Bruno, M. Conti, and E. Gregori, "Mesh networks: commodity multihop ad hoc networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123–131, 2005.

[18] A. Tonnesen, "OLSR: optimized link state routing protocol," Institute for Informatics at the University of Oslo (Norway). http://www.olsr.org.

[19] "AODV: ad hoc on demand distance vector routing," Department of Information Technology at Uppsala University (Sweden). http://core.it.uu.se/core/index.php/AODV-UU.

[20] F. Delmastro and A. Passarella, "An experimental study of p2p group-communication applications in real-world manets," in *Proceedings of IEEE ICPS Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality (REALMAN '05)*, Santorini, Greece, July 2005.

[21] F. Delmastro, A. Passarella, and M. Conti, "Experimental analysis of p2p shared-tree multicast on manets: the case of scribe," in *Proceedings of the IFIP 5th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net '06)*, Lipari, Sicily, Italy, June 2006.

[22] F. Delmastro, "From pastry to CrossROAD: CROSS-layer ring overlay for ad hoc networks," in *Proceedings of the IEEE 3rd International Conference on Pervasive Computing and Communications Workshops (PerCom '05)*, vol. 2005, pp. 60–64, Kauai Island, Hawaii, USA, March 2005.

[23] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, 2002.

[24] M. Castro, P. Druschel, A.-M. Kermarrec, and A. I. T. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489–1499, 2002.

[25] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Proceedings of the 3rd International Workshop on Networked Group Communication*, pp. 14–29, London, UK, November 2001.

[26] S. Q. Zhuang, B. Y. Zhang, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz, "Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination," in *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '01)*, Port Jefferson, NY, USA, June 2001.

[27] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Pittsburgh, Pa, USA, August 2002.

[28] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, vol. 37, no. 5, pp. 298–313, Lake George, NY, USA, October 2003.

[29] M. Ge, S. V. Krishnamurthy, and M. Faloutsos, "Application versus network layer multicasting in ad hoc networks: the ALMA routing protocol," *Ad Hoc Networks*, vol. 4, no. 2, pp. 283–300, 2006.

[30] C. Gui and P. Mohapatra, "Overlay multicast for MANETs using dynamic virtual mesh," *Wireless Networks Journal*, vol. 13, no. 1, pp. 77–91, 2007.

[31] E. Royer and C. Perkins, "Multicast ad hoc on-demand distance vector (maodv) routing," 2000. http://www3.ietf.org/proceedings/00jul/I-D/manet-maodv-00.txt.

[32] S.-J. Lee, W. Su, and M. Gerla, "On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks," 2000. http://www.cs.ucla.edu/NRL/wireless/PAPER/draft-ietf-manet-odmrp-02.txt.

[33] S. Yang and J. Wu, "New technologies of multicasting in MANET," in *Design and Analysis of Wireless Networks*, Y. Xiao and Y. Pan, Eds., Nova, Baltimore, MD, USA, 2005.

[34] J. Luo, P. Th. Eugster, and J.-P. Hubaux, "Probabilistic reliable multicast in ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 4, pp. 369–386, 2004.

[35] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location and routing for largescale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, vol. 2218 of *Lecture Notes in Computer Science*, pp. 329–350, Heidelberg, Germany, November 2001.

[36] FreePastry, Rice University. http://freepastry.rice.edu.

[37] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[38] E. Borgia, M. Conti, F. Delmastro, and L. Pelusi, "Lessons from an ad-hoc network test-bed: middleware and routing issues," *Ad Hoc & Sensor Wireless Networks, An International Journal*, vol. 1, no. 1-2, 2005.

[39] L. Ji and M. S. Corson, "Explicit multicasting for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 8, no. 5, pp. 535–549, 2003.