XScribe: a Stateless, Cross-Layer Approach to P2P Multicast in Multi-Hop Ad hoc Networks

Andrea Passarella
IIT-CNR
Via G. Moruzzi, 1
56124 Pisa, Italy
andrea.passarella@iit.cnr.it

Franca Delmastro
IIT-CNR
Via G. Moruzzi, 1
56124 Pisa, Italy
franca.delmastro@iit.cnr.it

Marco Conti
IIT-CNR
Via G. Moruzzi, 1
56124 Pisa, Italy
marco.conti@iit.cnr.it

ABSTRACT

Using p2p systems on multi-hop ad hoc networks is an amazingly interesting challenge. While the features of p2p systems designed for Internet are particularly suitable for ad hoc networking environments, the very assumptions behind their design are usually at odds with the ad hoc network distinctive features. It is thus challenging to port p2p features to ad hoc networks in an efficient way. In this work we focus on p2p multicasting, and design a Scribe cross-layer replacement, suitable for ad hoc networks (XScribe). We discuss the main XScribe features, and evaluate its performance on a real multi-hop ad hoc network. Results show that XScribe drastically improves the Scribe performance in terms of packet loss and delay. At the same time, they also indicate XScribe limitations, and suggest directions to further improve its design.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Design, Experimentation, Performance

Keywords

multi-hop ad hoc networks, p2p systems, multicast, cross-layer

1. INTRODUCTION

Multi-hop ad hoc networks and traditional p2p systems designed for the Internet share a number of common features. Both are completely self-organising, decentralised and self-healing. Both allow users to join and leave dynamically, with possibly high churn rates. Both are good platforms to host "p2p" applications in which nodes communicate directly with each other without any centralised service. It is thus extremely interesting to understand how features of legacy p2p systems can be ported to ad hoc networks. One of the big concerns is how to port them *efficiently*, because the typical trade-offs of legacy p2p systems are at odds with those of ad hoc network protocols. Usually, legacy p2p systems trade increased bandwidth consumption for higher scalability in terms of number of nodes, mainly because these systems should support even thousands of users. However, such large ad hoc networks are quite unlikely [13, 12]. At the same time, bandwidth is a scarce resource in ad hoc networking environments that has to be used sparingly.

In this paper we focus on p2p multicast, and consider a legacy solution made up of Pastry [16] and Scribe [5]. Pastry implements a Distributed Hash Table (DHT), while Scribe builds multicast trees on top of the DHT. Being mainly designed for handling exact-match queries, a structured overlay like Pastry 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 to 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, [4] 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.

We consider p2p multicast protocols instead of standard layer-3 multicast solutions. P2p multicast protocols exploit the indirection level provided by overlay networks, and run on the view of the network provided by the overlay. However, the literature on multicast on ad hoc networks takes into consideration both p2p and layer-3 approaches, and no clear winner has been identified yet [17]. In the field of ad hoc networks, there is actually an increasing trend towards blending together features usually implemented at the p2p and routing levels (e.g., [3, 7, 9]). Thus, we are working towards a multicast system within an integrated routing layer also including p2p features. Results of these work, and the new p2p multicast protocol we propose, can be seen as an helpful intermediate step towards this goal.

After recalling the main features of the solution based on Pastry and Scribe (Sections 2 and 3.1), we highlight the inefficiencies of such a p2p system when used as-it-is on ad hoc networks (Section 3.1.1). Based on these remarks, we propose a new solution that leverages cross-layer interactions between the p2p and the routing levels to optimise the p2p system implementation (see Figure 1). Specifically, we replace Pastry with CrossROAD [9] (Section 2), and Scribe with XScribe (Section 3.2). Both CrossROAD and XScribe provide to upper layers the same API defined by Pastry and Scribe, respectively, such that applications can be used unmodified on top of any of the two architectures. While CrossROAD has already proved to outperform Pastry on ad hoc networks (e.g., [2]), the main focus of this paper is the design and evaluation of XScribe. The main advantages of XScribe are: i) managing group membership with minimal overhead by exploiting the periodic traffic of a proactive routing protocol; and ii) delivering multicast data to intended receivers without requiring any networking structure (such as trees or meshes) built and maintained exclusively for multicasting purposes. In this sense, XScribe can be seen as a *stateless, cross-layer, p2p* multicast protocol.

In Section 4 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 XScribe is actually able to drastically improve the performance achieved by Scribe. In the majority of the cases, it is able to halve the packet loss and the delay *at the same time*. Despite its good performance, results also suggest XScribe limitations and ways to further improve it. We elaborate on these points both while discussing results, and while drawing conclusions in Section 5.

1.1 Related work

Pastry & Scribe is one of the platforms for p2p multicast designed for the wired Internet. Specifically, overlay networks similar to Pastry are also provided by CAN, Chord, Bamboo. Most of them can easily host p2p multicast protocols. However, we focused on Pastry and Scribe since they have shown to outperform these alternative approaches (see [6] and references herein).

Multicasting in ad hoc networks has received significant attention in the last few years. A number of solutions have been proposed, either implemented at the routing level (e.g., MAODV, ODRMP), or at the p2p (application) level (e.g., ALMA, PAST-DM). A good survey of them is provided in [17]. The closest approach to XScribe is the family of *stateless* protocols, such as DDM [14] and RDG [15]. With respect to these protocols, XScribe works at the p2p level, and leverages cross-layer interactions with a proactive routing protocol to improve the efficiency of the group-membership management.

This paper complements our previous work on p2p systems for multi-hop ad hoc networks. Specifically, in [10] we showed the advantage of replacing Pastry with CrossROAD, and using Scribe directly on top of it. In [11] we extensively evaluated the performance of Pastry and Scribe in ad hoc networks. In this paper we make a step further, and show how cross-layering can be leveraged to design a Scribe replacement optimised for ad hoc networks.

2. DHT: PASTRY VS. CROSSROAD

Pastry defines a DHT in a logical circular address space. The logical address of a node is the hashed value of its IP

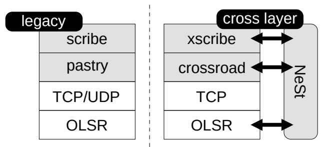


Figure 1: Network architectures.

address. A key is associated to each message sent on the overlay. 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.

The main costs of Pastry in terms of networking overhead are due to i) the overlay creation and management that require periodic communication between nodes, and ii) the multi-hop middleware routing caused by the incomplete knowledge of the overlay, 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. On the one hand, both theoretical [13] and experimental [12] papers show that large scale multi-hop ad hoc networks are not very likely due to intrinsic limitations of the wireless medium, and of the current wireless technologies. On the other hand, bandwidth is a very scarce resource on multi-hop ad hoc networks.

Based on these remarks, CrossROAD provides the same DHT features by taking a cross-layer optimised approach. Specifically, CrossROAD relies on a *proactive* routing protocol. A node wishing to join a CrossROAD overlay embeds few bytes into periodic routing packets, announcing its presence. This information eventually reaches all the other nodes in the network. Therefore, any node in the overlay knows the IP addresses of all the other nodes that compose the same overlay. The overlay creation can be done autonomously at each node by simply hashing these IP addresses. CrossROAD drastically reduces the bandwidth overhead with respect to Pastry [1, 2], because the cost of building the network is negligible. Furthermore, messages always travel just one-hop on the overlay (provided the overlay view is consistent at all nodes). In addition, experimental results [1] 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 [9]. As a final remark, note that all cross-layer interactions in our architecture are handled by the Network Status (NeSt) module [8], that allows to implement efficient cross-layer interactions without loosing the portability and maintainability of legacy layered stacks.

3. MULTICAST SYSTEMS

3.1 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 one

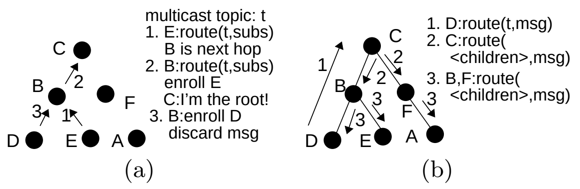

multicast topic: t
1. E:route(t,subs) B is next hop
2. B:route(t,subs) enroll E C:I'm the root!
3. B:enroll D discard msg
C
B
F
D
E
A
1
2
3
(a)
1. D:route(t,msg)
2. C:route(<children>,msg)
3. B,F:route(<children>,msg)
C
B
F
D
E
A
1
2
3
(b)

Figure 2: Scribe tree construction (a), and data delivery (b)

to the hashed topic. For example, in Figure 2 the root is node C. 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 Figure 2(a)), or discards the message otherwise (e.g., node B in step 3 in Figure 2(a)). In both cases, it enrolls the node from which it has 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 subsequently delivered by each parent to its children (steps 2 and 3 in Figure 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.

3.1.1 Limitations

We identify three main drawbacks in the Scribe design when Scribe is used on ad hoc networks. First, the data delivery mechanism is not optimised at all. Messages have firstly to reach the root, and are then delivered on the tree. This requires traversing several hops *on the overlay network*, which may lead to significant path stretches between a sender and each receiver. Second, Scribe tends to concentrate the burden of multicasting on a set of shared resources, most notably the links of the tree, and the root node. Our work in [11] shows that as the traffic load generated by multicast applications increases, the root node becomes overloaded, and the whole multicast group becomes unable to work properly. Third, the shared tree defined by Scribe has to be maintained. In wireless networks, where links are typically unstable, this might be costly, and may lead to tree partitions and nodes' isolation [10].

3.2 XScribe

To cope with the problems highlighted in Section 3.1.1, we propose XScribe, which is a replacement for Scribe optimised through cross-layer interactions. XScribe is heavily inspired by *stateless, explicit* multicast approaches such as DDM [14] and RDG [15]. For ease of explanation, we divide the XScribe operations into *data dissemination* and *membership management*, and discuss each aspect separately.

3.2.1 Data dissemination

As in DDM, in XScribe each sender of a multicast group is aware of all the other group members. Specifically, in XScribe each sender keeps a list with the logical addresses (in the overlay network) of the group members (section 3.2.2 shows how this is achieved). Locally generated messages are disseminated in the group by sending to each member a distinct message over the overlay network.

Even though this data-dissemination mechanism can be further optimised, it addresses all the points we have raised in Section 3.1.1. Firstly, it avoids possibly long path stretches. If we pick a particular sender-receiver pair, in XScribe messages are sent directly to the receiver from the sender, without having to go through an intermediate root node as in Scribe. Thanks to CrossROAD routing, messages actually travel along the best possible path between the sender and the receiver, according to the underlying L3 routing protocol. Secondly, XScribe does not require any root node, and it spreads the burden of multicasting more evenly than Scribe does, since messages are exchanged in a pure peer-to-peer fashion from producers to consumers. Thirdly, being completely stateless, it does not require any procedure to maintain structures exclusively related to multicasting, such as trees or meshes.

Despite these advantages, it is easy to show that the XScribe dissemination policy could be further optimised, as discussed in Section 5. The repeated unicast used by XScribe is definitely a feature to be improved. However, it is easy to show that, in small-scale ad hoc networks, Scribe tends to build two-level trees in which all 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 current version of XScribe has the advantage of being extremely simple and straightforward, and, as shown in Section 4, it already drastically improves the performance of the legacy Scribe solution.

3.2.2 Membership management

XScribe uses a cross-layer policy for managing membership, which is inspired by the way CrossROAD works. Also in this case, the main idea is exploiting the periodic 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 group (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 members of available multicast groups, which is 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 will not be able to join multicast groups, while they will still be able to run other applications based on CrossROAD.

Essentially, this is the same mechanism CrossROAD uses to advertise nodes' presence in the overlay, and is efficient from a bandwidth overhead standpoint. For example, if the bitmask size is 128 bits, and the period used by the routing protocol to send updates is 2 seconds (the default value for OLSR), the overhead originated by a XScribe node is just

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

8 Bps. Furthermore, XScribe information does not need to be sent in separate frames at the MAC level, and does not require additional accesses to the shared medium. This is very important whenever the network starts to be even slightly congested.

Conceptually, this membership management policy could be implemented also in layer-3 multicast systems, and does not necessarily require the presence of a DHT. However, in this work we have chosen to keep XScribe at the p2p level, as the original Scribe is, to have a direct comparison among the two systems. Also note that the policy to manage membership is one of the main differences between XScribe and other stateless multicast protocols such as DDM or RDG. Both DDM and RDG require each receiver to send a join 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.

4. PERFORMANCE EVALUATION

4.1 Evaluation scenario

To compare the legacy and the cross-layer multicast p2p systems we have run experiments in a real ad hoc network setup. Specifically, we have implemented CrossROAD and XScribe, while we have used the Pastry and Scribe implementations provided by the Rice University (FreePastry). To have a realistic application environment, we have also implemented a simple Whiteboard Application (WB), 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 to, and start drawing on a canvas, also receiving drawings of the other users. All the nodes whose users subscribe to the same topic are joined together by the p2p multicast protocol. The software implementing the protocol stacks can be downloaded from
`http://bruno1.iit.cnr.it/xscribe_exp/`.

The testbed we have used represents a small-scale multihop ad hoc network, as shown in Figure 4. All nodes were IBM ThinkPad R50 laptops with 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. Nodes A to E ran the whole protocol stacks, including the WB application, while nodes R1 and R2 just acted as routers. Multi hopping in our testbed was actually emulated by using `iptables`, and we cross-checked that paths were actually multi-hop by inspecting packet traces collected during the experiments. Although not able to completely capture all effects of wireless links' intricacies, this setup allowed us to closely approximate the system behavior in a realistic multi-hop setting. We did not run experiments in mobile conditions, so as to decouple the effects of mobility and architectural differences. Analysing the system in different topologies and in mobile scenarios are main subjects of future work.

In this particular configuration, nodes were always just one-hop away from each other also in the Pastry overlay. This is a favourable condition for Pastry, which actually makes its behavior closer to the CrossROAD one. Thus, the performance differences that we highlight in the following are mainly related to the different design of the multicast systems.

Users on nodes A to E joined the same topic at the beginning of each trial, and remained subscribed for the whole duration of the trial. In the case of Scribe, node C was the root of the multicast tree. Finally, in our tests just users at nodes C and D generated traffic. This configuration minimises the traffic related to XScribe membership management. Therefore, in the following we just focus on the performance gains achieved by improving the data dissemination policy.

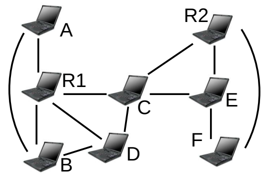


Figure 4: Experiment topology

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 receiving other users' strokes. Both ON and OFF phase lengths were exponentially distributed. Each trial was composed by 100 active/idle cycles, and in any configuration 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 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 a cycle, and the average duration of an active/idle cycle. We have found that this simple index is sufficient to correctly identify usage cases for our scenario.

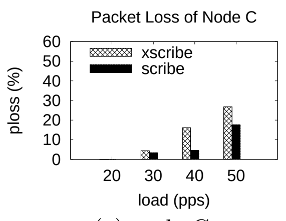
We characterise the architectures' performance at each node in terms of packet loss and delay statistics. Specifically, the packet loss at node i is measured as $1 - \frac{R_i}{\sum_{j=1}^{N} S_j}$, where R_i is the number of messages received by node i, N the number of senders, and S_j is the number of messages transmitted by the j-th sender. Packet delays are measured by timestamping the transmission time at the sender, and reception time at the receiver (recall that nodes are synchronised). Finally, for nodes acting as senders, we did not take into account locally generated packets to compute the performance figures.

We replicated each configuration several times, obtaining quite variable results. They are mainly due to the characteristic variability of the wireless medium. In this paper we show the best results measured in each configuration.

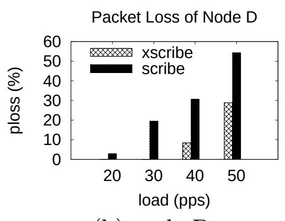
4.2 Packet loss

Figure 3 shows the packet loss experienced by each node under the two alternative architectures, as a function of the application load (loads below 20pps 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

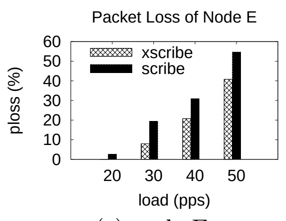
[2]A distinct message was sent for each stroke. The size of each message was 1448 bytes.



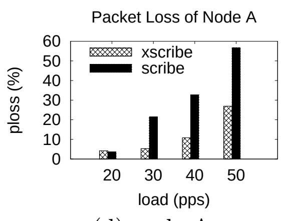
(a) node C



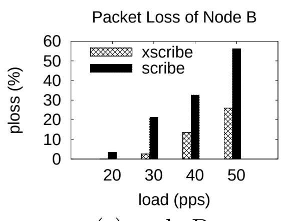
(b) node D



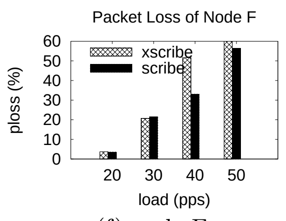
(c) node E



(d) node A



(e) node B



(f) node F

Figure 3: Packet loss at each node

occurs when these queues fill up, and is thus a side effect of network congestion[3].

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 details, at nodes D, A, and B the packet loss is always more than halved with respect to the Scribe cases. Different observations should be drawn with respect to node C (the Scribe root), and node F. As far as node C, the packet loss under XScribe is reasonably low, and similar to the packet loss of other nodes in the center of the network. This had to be expected, as in XScribe node C does not play any special role. What is surprising is the fact that the root node in the Scribe case achieves lower packet loss than the same node in the XScribe case. We are actually working to get more precise insights on this, but this particular result seems to indicate a general limitation of XScribe. By looking more carefully at TCP-connection traces, we have found that some TCP connections might get suddenly "frozen" for some time. These phenomena are slightly more frequent on XScribe than in Scribe, and can justify the lower performance in terms of packet loss experienced by nodes C and F in this particular case. A possible reason for such freezing events could be the fact that XScribe tends to generate more TCP connections than Scribe does. We are actually working to precisely figure out the effect of unfairness between concurrent TCP connections on XScribe. However, note that in the majority of the cases the XScribe disadvantage of generating more TCP connections is overcome by far by the advantage of its simpler architectural design.

Overall, XScribe appears to scale better than Scribe with the traffic load. Therefore, XScribe allows the application to operate at higher loads than Scribe does, whatever usability threshold the application might define with respect to the maximum acceptable packet loss.

[3]Properly dimensioning the queues 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 queues to grow unlimited).

4.3 Delay

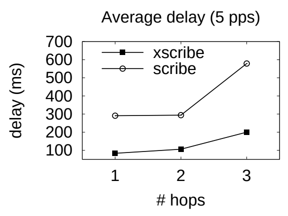
Table 1 shows the average values and the 90th percentiles of delay experienced by each node. Three traffic loads have been selected, representative for light, medium, and high loads. Results are almost self-explanatory. When XScribe is used, the average delay is more than halved, while the 90th percentile is reduced by at least 1.5x. Coupled with the result related to packet loss, this tells 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*, reduce the average delay. As far as nodes C and F, we can see that XScribe outperforms Scribe, which is the opposite of what measured in terms of packet loss. This highlights that, apart from the phases during which TCP gets frozen causing bursts of packet losses, XScribe is more efficient than Scribe.

As a final remark about delay performance, Figure 5 shows the average delays between nodes that were *physically* 1 hop, 2 hops and 3 hops away from each other, for different traffic loads[4]. On the one hand, these plots tell which delay a user might expect, depending on its distance with respect to a sender. On the other hand, they show the negative effect of the higher centralisation and the path stretch brought by the Scribe architecture (recall that in our configuration nodes are one hop away also in the Pastry overlay). It is also interesting to note that, in the case of Scribe, these figures depend on the particular position of the root node in the network topology. Actually, if the root node is placed at one edge of the network, the delay performance can be far worse than that (see [11]). Clearly, XScribe is immune to this problem.

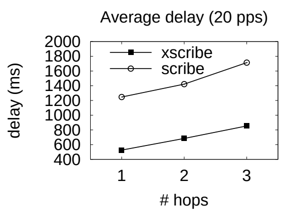
5. CONCLUSION & PERSPECTIVE

In this work we have presented XScribe, which is a simple cross-layer replacement for Scribe designed for multi-hop ad hoc networks. The main ideas behind XScribe are i) exploiting a proactive routing protocol to manage and disseminate nodes' membership information in the network, and ii) use nodes' membership information to send multicast data with-

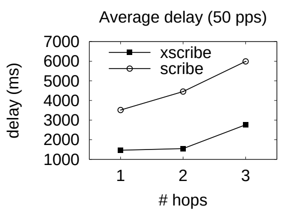
[4]Similar trends also occurred for the 90th percentiles. These results are omitted here for the sake of space.



(a) 5 pps



(b) 20 pps



(c) 50 pps

Figure 5: Average delay as a function of the hop count

Table 1: Delay statistics at each node (in ms)

node	load (pps)	average		90th percentile	
		Scribe	XScribe	Scribe	XScribe
C	5	264	132	622	219
	20	714	444	1660	1050
	50	2200	1290	3390	2830
D	5	317	77	688	167
	20	1650	594	3190	1270
	50	4590	1570	6900	3630
E	5	310	123	674	242
	20	1350	556	2690	1140
	50	4620	1950	6900	4220
A	5	307	120	686	224
	20	1420	612	2940	1340
	50	4810	1350	7140	2820
B	5	356	28	721	224
	20	1480	624	3110	1400
	50	4710	1400	7050	3200
F	5	318	164	687	315
	20	1610	884	3200	1910
	50	4760	2780	7100	6190

out requiring any networking structure specifically devoted to multicasting (such as a tree or a mesh).

Results presented in this work clearly indicate that data dissemination is much more efficient when such a solution is adopted instead of the original Scribe approach. Specifically, XScribe is able, in the majority of the cases, to halve the packet loss and the average delay *at the same time.*

Nevertheless, our results also show limitations of the current XScribe design. Data dissemination could be furhter optimised by exploiting partially shared paths among sets of multicast receivers (as traditional L3 multicast does). Moreover, the number of TCP connections required by XScribe tends to be too high for ad hoc networks. This is the price paid in order to keep the XScribe design straightforward, and make it operate on top of generic DHTs. A possible direction to address these problems could be exploiting more information available at the routing layer to further optimise the XScribe delivery policy. Furthermore, the subject-based routing features of DHTs could be exploited to reduce the membership management overhead. We are actually working to improve XScribe along these directions.

6. ACKNOWLEDGEMENTS

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

7. REFERENCES

[1] 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 *Proc. of the ACM SIGCOMM E-WIND Workshop*, Philadelphia, Aug. 2005.

[2] E. Borgia, M. Conti, F. Delmastro, and L. Pelusi. Lessons from an Ad-Hoc Network Test-Bed: Middleware and Routing Issues. In *Ad Hoc & Sensor Wireless Networks, An International Journal*, Vol.1, Numbres 1-2, 2005.

[3] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: Network routing inspired by dhts. In *Proc. of ACM SIGCOMM*, Pisa, Italy, September 11-15 2006.

[4] M. Castro, M. Costa, and A. Rowstron. Debunking some myths about structured and unstructured overlays. In *Proc. of NSDI*, May 2-5 2005.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralised application-level multicast infrastructure. *IEEE JSAC*, 20(8), Oct. 2002.

[6] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An evaluation of scalable application-level multicast built using peer-to-peer overlays. In *Proc. of INFOCOM*, San Francisco, CA, Apr. 2003.

[7] M. Conti, E. Gregori, and G. Turi. Towards scalable p2p computing for mobile ad hoc networks. In *Proc. of the second IEEE Annual Coneference on Pervasive Computing and Communications Workshops*, Orlando, Mar. 2004.

[8] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross layering in mobile ad hoc network design. *IEEE Computer*, Feb. 2004.

[9] F. Delmastro. From Pastry to CrossROAD: Cross-layer Ring Overlay for AD hoc networks. In *Proc. of IEEE PerCom MP2P Workshop*, Kauai Island, Hawaii, Mar. 2005.

[10] F. Delmastro and 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.

[11] F. Delmastro, A. Passarella, and 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.

[12] P. Gunningberg, H. Lundgren, E. Nordstroem, and C. Tschudin. Lessons from experimental manet research. *Ad Hoc Networks Journal*, 3(2):221–233, Mar. 2005.

[13] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, Mar. 2000.

[14] L. Ji and M. Corson. Explicit multicasting for mobile ad hoc networks. *Mobile Networks and Applications*, 8:525–549, 2003.

[15] J. Luo, P. Eugster, and J.-P. Hubaux. Probabilistic reliable multicast in ad hoc networks. *Ad Hoc Networks*, 2:369–386, 2004.

[16] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object localtion and routing for large scale peer-to-peer systems. *LNCS*, 2218:329–350, 2001.

[17] S. Yang and J. Wu. *New Technologies of Multicasting in MANETS.* Nova Publishers, 2005.