

On developing P2P Group-Communication Applications in Real-World MANETs: and Experimental Study*

Franca Delmastro

CNR, IIT Institute

Via G. Moruzzi, 1 – 56124 Pisa, Italy

franca.delmastro@iit.cnr.it

Andrea Passarella

University of Cambridge, The Computer Laboratory

15 JJ Thomson Avenue – Cambridge CB3 0FD, UK

andrea.passarella@cl.cam.ac.uk

Abstract

Group-communication applications are a very promising opportunity for developing valuable MANET-based applications. However, real-world experimental studies are required to indicate the best solutions to implement them. We have implemented a real prototype in which alternative networking stacks can be used to support a distributed Whiteboard application. By means of experimental results we show that a standard P2P solution based on Pastry and Scribe is not suitable for MANET environments. We also show that a cross-layer P2P system optimised for MANETs (i.e., CrossROAD) is able to overcome many of the problems experienced with Pastry.

1 Introduction

Even though research on MANETs has been very active in the last decade, real applications addressed to people outside the research community still have to be developed. The typical simulation-based approach for the performance evaluation of MANETs is one of the main reasons of this. Often, simulation results turn out to be quite unreliable if compared to real-world measurements [1, 7], and real-world experiments are highly required for MANET applications to become reality, despite their high costs (in terms of time to set up) and intrinsic limitations (number of nodes).

By leveraging the self-organising nature of MANETs, group-communication applications can be an outstanding opportunity from this standpoint. In this paper, we focus on a significant example of this class of applications, and we evaluate complete networking solutions that could be used to develop it. Specifically, we consider the Whiteboard application (WB), which implements a distributed whiteboard among MANET users. WB allows users to share drawings, messages and other dynamically generated contents. Such a self-organising distributed application can be naturally supported by P2P systems. In our prototype, WB uses Scribe [2] to share WB contents among users via application-level multicast trees. Scribe requires a P2P overlay network

based on a DHT. Our prototype includes two alternative P2P solutions, i.e., Pastry [8] and CrossROAD [5]. Both of them provide the same functionalities toward above layers through the P2P commonAPI [11], but CrossROAD is explicitly designed for MANET environments. It reduces (with respect to Pastry) the network overhead related to the overlay management by exploiting cross-layer interactions with a proactive routing protocol. Specifically, CrossROAD implementation is compliant to the cross-layer framework described in [4]. Finally, the prototype includes OLSR [9] and AODV [10] at the routing level. Pastry performances are evaluated on top of both routing protocols while CrossROAD is evaluated on top of OLSR, since CrossROAD is designed to exploit a cross-layer interaction with a proactive routing protocol.

The main contribution of this paper is evaluating through real experiments *complete* networking solutions for developing distributed applications such as WB in real-world MANETs. We evaluate our prototype at two different levels, i.e., we quantify i) the QoS perceived by WB users, and ii) the quality of the multicast tree generated by Scribe. First of all, we show how a proactive routing protocol performs better than a reactive one with regard to this kind of applications. Then, we highlight that a solution based on Pastry and Scribe is not suitable for MANET environments. WB users perceive unacceptable high data loss and delays. Furthermore, both the Pastry overlay network and the Scribe multicast tree get frequently partitioned. This results in some WB users to be completely isolated from the rest of the network. Finally, we show that some of these problems can be avoided by using CrossROAD. Specifically, the structure of the Scribe tree is quite more stable when CrossROAD is adopted, and partitions problems experienced with Pastry completely disappear. Thus, CrossROAD turns out to be a very promising P2P system for MANET environments.

2 WB and its middleware support

The Whiteboard application was originally designed in [6], and then we adapted it to work on top of Pastry and CrossROAD. It implements a distributed whiteboard, that

*This work was partially funded by the FET-IST Programme of the European Commission, IST-2001-38113 MOBILE-MAN project.

can be used to share dynamically generated contents (e.g., drawings, messages, ...). Each user runs a WB instance on her mobile device, and selects a *topic* she wants to associate to (i.e. “treasure hunting”). Each topic is linked with a canvas on which she can draw strokes or type text. On the same canvas, the user directly sees strokes and text generated by others. Being a simple example of group-communication applications, WB allows us to understand how such applications can be successfully developed in MANETs.

WB needs a subject-based multicast protocol to build groups (i.e., identify all nodes whose users are interested into the same topic), and disseminate WB data to the group members. Specifically, in our testbed, Scribe [2] is used as the multicast protocol, since it has shown to outperform other similar solutions [3]. Scribe is designed to work on top of Pastry, but it can be used with any P2P system providing a commonAPI-compliant overlay network, such as CrossROAD.

2.1 Pastry and CrossROAD

Pastry is a P2P system based on a DHT to build a structured overlay network (*ring*) at the middleware level. A logical identifier (node id) is assigned to each node hashing one of its physical identifiers (e.g., IP address, hostname). Messages are sent on the overlay by specifying a destination key k belonging to the logical identifiers’ space. Pastry routes these messages to the node whose id is numerically closest to k value. To route messages, Pastry nodes maintain a limited subset of other nodes’ logical ids in their internal data structures (middleware routing tables). Periodic data exchange between nodes of the overlay are needed to update the state of the overlay. Finally, in order to initially join the overlay network, each Pastry node executes a bootstrap procedure, during which it initialises its middleware routing table by collecting portions of other nodes’ routing tables. Specifically, each node has to connect to an already existing Pastry node (i.e., it needs to know its IP address) in order to correctly start the bootstrap procedure.

The bootstrap phase and the periodic data exchange between nodes constitute the main network overhead of Pastry. CrossROAD, that is a Pastry-like P2P system explicitly designed for MANETs, drastically reduces the Pastry overhead by exploiting cross-layer interactions with a proactive routing protocol. Specifically, CrossROAD defines a cross-layer Service Discovery protocol in order to broadcast information about upper-layer services (e.g. Scribe) through the proactive flooding of routing packets, and to maintain an association between nodes’ IP addresses and provided services. Hence, each CrossROAD node can autonomously build the overlay, by simply hashing the IP address of nodes providing the same service. In this way the overlay network related to a particular service is maintained with almost *negligible network overhead* in compari-

son with Pastry. Furthermore, CrossROAD i) is completely self-organising, since it does not require any bootstrap procedure, and ii) correctly manages cases of network partitioning and topology changes with the same delays of the routing protocols.

2.2 Scribe

Scribe exploits Pastry-like routing to build multicast groups. From the standpoint of the application running on Scribe, the group is identified by a *topic*. Scribe uses the hash function provided by Pastry (or CrossROAD) to generate the topic id (t_{id}) in the logical space of node ids. In order to join the Scribe tree, nodes send a `join` message on the overlay with key equal to t_{id} . This message reaches the next hop (say, N) towards the destination on the overlay network. The node originating the `join` message is enrolled as a child of N . If not already in the tree, N itself joins the tree by generating a `join` message anew. Eventually, such a message reaches the node whose id is the closest one to t_{id} and is not propagated further. This node is defined as the *root* of the Scribe tree.

Application messages are sent on the overlay with key equal to t_{id} . Hence, they reach the Scribe root, which is in charge of delivering them over the tree. To this end, it forwards the messages to its children, which further forward them to their children, and so on.

Finally, the Scribe maintenance procedure is as follows. Each parent periodically sends a `HeartBeat` message to each child¹. If a child does not receive any message from the parent for a given time interval (20 s in the default case), it assumes that the parent has given up, and re-executes the join procedure. This simple procedure allows node to discover parent failures, and re-join the tree, if the case.

3 Experimental Environment

The experiments reported in this paper are based on a *static* MANET. This allows us to highlight limitations that originate from Pastry and Scribe design, rather than to mobility. Extending the results in the case of mobility is subject of future work.

The experiment testbed is as depicted in Figure 1. We set up an indoor MANET consisting of 8 nodes. To have an homogeneous testbed, all nodes are IBM ThinkPad R50 laptops. We use the built-in Intel PRO-Wireless 2200 802.11 card, with `ipw2200` driver (on Linux 2.6 kernel). The data rate is set to 11 Mbps. In addition the transmission power of each card has been adjusted to reproduce the topology shown in the figure and obtain a multi-hop ad hoc network. During the experiments, nodes marked A through to F participate in the overlay network, and run the WB application (they will be throughout referred to as “WB nodes”).

¹Application-level messages are used as implicit `HeartBeats`.

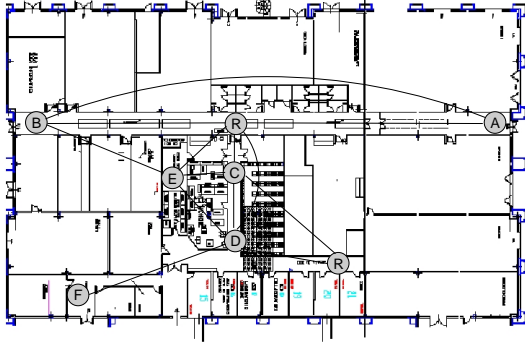


Figure 1. Map of the experiment setup

Nodes marked with “R” are used just as routers. It is worth pointing out that this setup lies within the “802.11 ad hoc horizon” envisioned in [7], i.e. 10-20 nodes, and 2-3 hops. Therefore, it is a valid example of possible real-world MANETs.

In order to have a controllable and reproducible setup, a human user at a WB node is represented by a software agent running on the node. During an experiment, each software agent interleaves active and idle phases. During an active phase, it draws a burst of strokes on the canvas, which are sent to all the other WB nodes through Scribe². During an idle phase, it just receives possible strokes from other WB nodes. After completing a given number of such *cycles* (a cycle is defined as a burst of strokes followed by an idle time), each agent sends a `Close` message on the Scribe, waits for getting `Close` messages of all the other nodes, and shuts down. Burst sizes and idle phase lengths are sampled from exponentially distributed random variables. The average length of idle phases is 10 s, and is fixed through all the experiments. On the other hand, the average burst size is defined on a per-experiment basis. As a reference point, we define a traffic load of 100% as the traffic generated by a user drawing, on average, one stroke per second. Finally, the number of cycles defining the experiment duration is fixed through all the experiments. Even at the lowest traffic load taken into consideration, each agent draws – on average – at least 50 strokes during an experiment. For the performance figures defined in this paper (see below) this represents a good trade-off between the experiment duration and the result accuracy.

Some final remarks should be pointed out about the experiment start-up phase. Nodes are synchronised at the beginning of each experiment. Then, in the Pastry case, the Pastry bootstrap sequence occurs as follows³: node C starts

²Please note that in our experiment each stroke generates a new message to be distributed on the Scribe tree.

³The same schedule is also used to start CrossROAD, even though a CrossROAD node does *not* need to bootstrap from another node.

first, and generates the ring. Nodes E and D start 5 seconds after C, and bootstrap from C. Node B starts 5 seconds after E and bootstraps from E. Node A starts 5 seconds after B and bootstraps from B. Finally, node F starts 5 seconds after D and bootstraps from D. After this point in time, the Scribe tree is created and, finally, WB instances start sending application messages (hereafter, WB messages). This way, the Scribe tree is built when the overlay network is already stable, and WB starts sending when the Scribe tree is completely built.

3.1 Performance Indices

Since Pastry and Scribe have been conceived for fixed networks, we investigate if they are able to provide an adequate Quality of Service to users in a MANET environment. To quantify the “WB user satisfaction” we use two performance indices:

Packet Loss: at each node i , we measure the number of WB messages received and sent (R_i and S_i , respectively) during an experiment; the packet loss experienced by node i is defined as $pl_i = \frac{R_i}{\sum_i S_i}$.

Delay: the time instant when each packet is sent and received is stored at the sending and receiving node, respectively. This way, we are able to evaluate the delay experienced by each node in receiving each packet. If d_{ij} is the delay experienced by node i in receiving packet j , and N_i the total number of packets received by i during an experiment, the average delay experienced by node i is defined as $D_i = \frac{\sum_j d_{ij}}{N_i}$.

Furthermore, we define two more indices, to quantify the quality of the multicast tree created by Scribe.

Node Stress: for each node, it is defined as the average number of children of that node. If t_{ij} is the time interval (within an experiment) during which node i has n_j children, the average node stress of node i is $NS_i = \frac{\sum_j n_j t_{ij}}{\sum_j t_{ij}}$.

Re-subscriptions: for each node, we count the number of times (during an experiment) this node sends new subscriptions requests, because it can’t communicate with the previous parent anymore.

4 Performance with Pastry

The results we report in this section are obtained by using Pastry as DHT, and either OLSR or AODV as routing protocol. Experiments are run by increasing the traffic load starting from 20% up to 80%.

Before presenting the results in detail, let us define what hereafter will be referred to as “crash of the Scribe Root Node”. In our configuration Pastry assigns node ids by hashing the IP address and the port used by Scribe on the

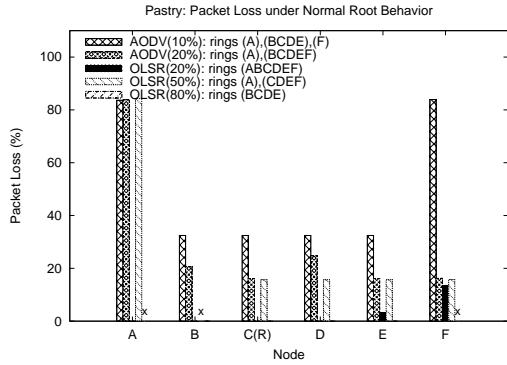


Figure 2. Packet Loss w/o MSRN crash

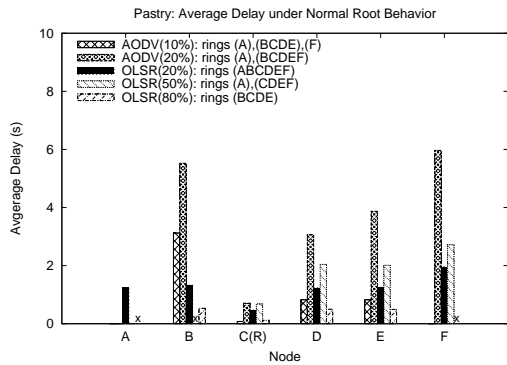


Figure 3. Delay w/o MSRN crash

node. Hence, each node always gets the same node id. Furthermore, the topic used by the WB users is always the same. Under the hypothesis that Pastry generates a single ring encompassing all WB nodes, the Root of the Scribe tree (i.e., the node whose id is closest to the WB topic id) is the same through all the experiments, and is node C in Figure 1. This node will be throughout referred to as the Main Scribe Root Node (MSRN). Due to the Scribe algorithm, each WB message to be distributed on the tree is firstly sent to MSRN, and then forwarded over the tree. Often, this is an excessive load for MSRN, which, after some point in time, becomes unable to deliver all the received messages. Instead, messages are dropped at the MSRN sending queue. We refer to this event as a crash of MSRN. Of course, since the application-level traffic is randomly generated, the MSRN crash is not a deterministic event.

4.1 User Satisfaction

Figures 2 and 3 show the packet loss and the delay indices experienced by the WB nodes considering experiments where MSRN *does not* crash. Specifically, we consider AODV experiments with 10% and 20% traffic load, and OLSR experiments with 20%, 50% and 80% traffic

load, respectively. There is no point in running AODV experiments with higher traffic load, since performances with AODV are quite bad, even with such a light traffic load. In the figure legend we also report the rings that Pastry builds during the bootstrap phase (please note that, theoretically, just one ring should be built, encompassing all WB nodes). Finally, 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 (for example, because some component of the stack crashed during the experiment).

Figure 2 allows us to highlight an important Pastry weakness. If a WB node is unable to successfully bootstrap, it starts a new ring, and remains isolated for the rest of the experiment. In MANET environments, links are typically unstable, and the event of a WB node failing to contact the bootstrap node is quite likely. Clearly, once a node is isolated, it is unable to receive (send) WB messages from (to) other nodes for the rest of the experiment, and this results in packet losses at all nodes. In the “AODV 10%” experiment, nodes A and F are isolated, and create their own rings. This results in packet loss of about 80% at those nodes (i.e., they just get their own WB messages, which is about one sixth of the overall WB traffic), and about 33% at nodes B, C, D and E. Similar remarks apply to the “OLSR 50%” experiment. It is more interesting to focus on the “AODV 20%” experiment. In this case, node A is isolated, while nodes B, C, D, E and F belong to the same ring. As before, A’s packet loss is about 80%. The packet loss at the other nodes due to the isolation of node A is about 18% (one sixth of the overall traffic). It is interesting to notice that nodes B and D experience a *higher* packet loss, meaning that they are unable to get WB messages generated within the “main” Pastry ring (i.e., nodes B, C, D, E, F). Finally, in the case “OLSR 20%”, Pastry is able to correctly generate a single ring, and the packet loss is quite low. In the case “OLSR 80%” nodes A and F crash. However, the packet loss experienced by the other nodes is negligible.

Similar observations can be drawn by focusing on the delay index (Figure 3). First of all, it should be pointed out that the delay related to nodes that are the sole member of their own ring (e.g., node A in the “AODV 10%” case) is obviously negligible. Even though – in general – the delay in this set of experiments is low, it can be noted that better performances are achieved by using OLSR instead of AODV. Finally, it should be noted that MSRN (node C) always experiences a lower delay with respect to the other nodes in the same ring.

Figures 4 and 5 show the packet loss and the delay indices in cases of MSRN crash. The packet loss experienced by nodes in the *same* ring becomes higher than in cases where MSRN does not crash. In the first three experiments, node A isolation causes a packet loss of about 18% on the

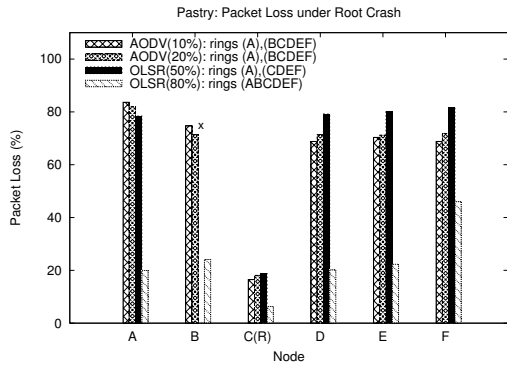


Figure 4. Packet Loss w/ MSRN crash

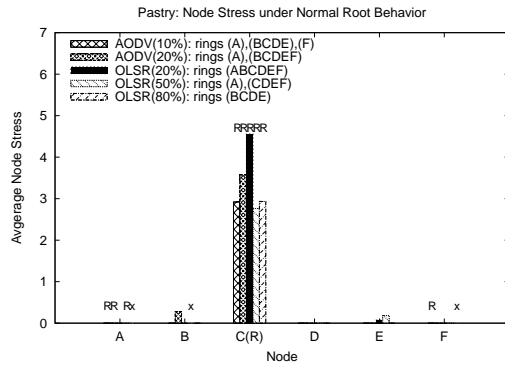


Figure 6. Node stress w/o MSRN crash

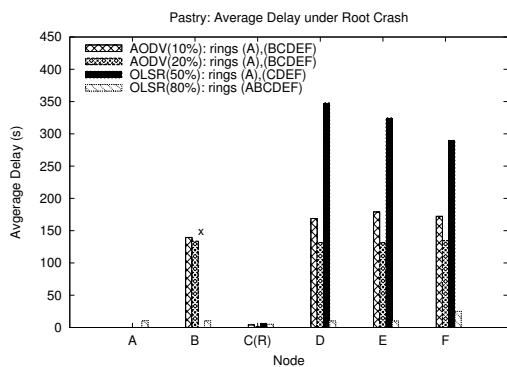


Figure 5. Delay w/ MSRN crash

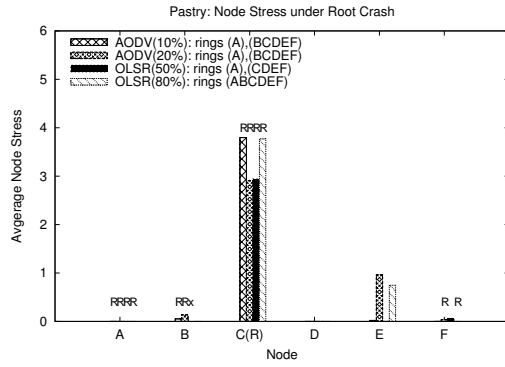


Figure 7. Node stress w/ MSRN crash

other nodes. Hence, MSRN crash is in charge of the remaining 60% packet loss. Quite surprisingly, OLSR with 80% traffic load shows better performance than OLSR with 50% traffic load. It is also interesting to note that the packet loss at MSRN is always lower than at other nodes in the same ring. This highlights that MSRN is able to get, but unable to *deliver* over the Scribe tree WB messages generated by other nodes. Similar observations can be drawn by looking at Figure 5, as well. The delay experienced by nodes B, D, E and F can be as high as few *minutes*, either by using AODV or OLSR. Finally, the delay experienced by MSRN is very low in comparison to the delay experienced by the other nodes.

To summarise, the above analysis allows us to draw the following observations. The Pastry bootstrap algorithm is too weak to work well in MANETs, and produces unrecoverable partitions of the overlay network. This behavior is generally exacerbated by AODV (in comparison to OLSR). Furthermore, MSRN is clearly a bottleneck for Scribe. MSRN may be unable to deliver WB messages also with moderate traffic loads, resulting in extremely high packet loss and delay. Moreover, the performance of the system in terms of packet loss and delay is unpredictable.

With the same protocols and traffic load (e.g., OLSR and 50% traffic load), MSRN may crash or may not, resulting in completely different performance figures. In cases where MSRN crashes, packet loss and delay are clearly too high for WB to be actually used by real users. However, even when MSRN does not crash, the high probability of WB users to be isolated from the overlay network makes Pastry-based solutions too unreliable. These results suggest that Pastry and Scribe need to be highly improved to actually support group communication applications such as WB in MANET environments.

4.2 Multicast Tree Quality

In this section we analyse the node stress and re-subscription indices, with respect to the same experiments used in the previous section.

Figures 6 and 7 plot the average node stress with and without MSRN crashes, respectively. In both cases, the node stress is significantly higher at MSRN than at any other node. This means that the Scribe tree is a one-level tree, and MSRN is the parent of *all* the other nodes. This behavior is expected, and can be explained by recalling the way Scribe works. In our moderate-scale MANET, all

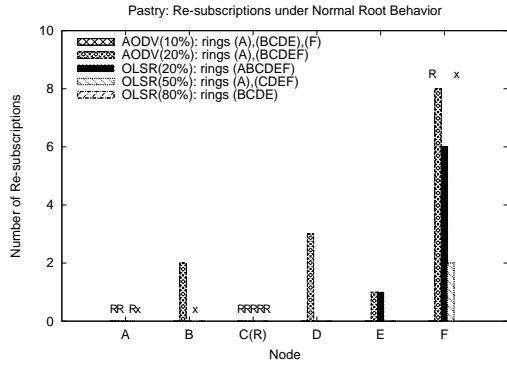


Figure 8. Re-subscriptions w/o MSRN crash

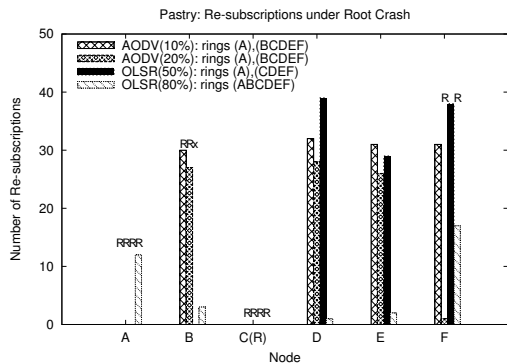


Figure 9. Re-subscriptions w/ MSRN crash

nodes are in the Pastry routing table of each other. Hence, Scribe join messages reach MSRN as the first hop, and MSRN becomes the parent of all other nodes (in the same ring). Together with the way application-level messages are delivered, this phenomenon explains why MSRN is a bottleneck, since it has to send a distinct message to *each* child when delivering WB messages over the tree. This is a major limitation of the Scribe algorithm, and optimisations of the P2P system are clearly not sufficient to cope with it.

In Figures 6 and 7 we have added “R” labels to indicate nodes that occur to become Scribe Root during the corresponding experiment. When MSRN does not crash (Figure 6) other nodes become Scribe root only as a side effect of a failed Pastry bootstrap. On an isolated WB node, Scribe builds a tree which consists only of the node itself, that is thus the root. However, Scribe partitions may also occur due to congestion at the Pastry level in cases where MSRN crashes. By looking at Figure 7, it can be noticed that nodes other than MSRN may become root also if they belonged (after the Pastry bootstrap phase) to the same overlay network of MSRN. This phenomenon occurs, for example, at node A in the OLSR 80% case, and at node B and F (whenever they become root). It should be noted that a

node with id n_1 (other than MSRN) becomes root when i) it loses its previous parent, and ii) the Pastry routing table does not contain another node id n_2 such that n_2 is closer to the WB topic id than n_1 . Figure 7 shows that the congestion at the Pastry level is so high that the Pastry routing table of some nodes becomes incomplete (i.e., MSRN disappears from other nodes’ routing table). Thus, the Scribe tree gets partitioned in several isolated sub-trees. Clearly, this contributes to the high packet loss measured in these experiments. Another effect of Pastry congestion during MSRN crashes is a possible reshaping of the Scribe tree. Figure 7 shows that the average Node Stress of E is close to 1 in the “AODV 20%” and “OLSR 80%” cases. This means that MSRN disappears from the Pastry routing table of some node, which – instead of becoming a new root – finds node E to be the closest one to the WB topic id. This phenomenon could be considered a benefit, since it reduces the MSRN node stress. However, it derives from an incorrect view of the network at the Pastry level, originated from congestion.

Figures 8 and 9 show the re-subscription index for the same set of experiments. Figure 8 shows that, when MSRN does not crash, the Scribe tree is quite stable. Most of the re-subscriptions occur at node F, which is the “less connected” node in the network (see Figure 1). In these experiments, the performance in the AODV cases is worse than in OLSR cases. Furthermore, upon MSRN crashes (Figure 9), the number of re-subscriptions increases drastically, even in case of “well-connected nodes” (i.e., node B, D and E). MSRN crashes make other nodes unable to get messages from their parent (i.e., MSRN itself), increasing the number of re-subscriptions. It is interesting to point out that this is a typical positive-feedback control loop: the more MSRN is congested, the more re-subscriptions are sent, the more congestion is generated.

To summarise, the multicast tree generated by Scribe on top of Pastry is quite unstable, especially in cases of MSRN crashes. The tree may get partitioned in disjoint sub-trees, and many re-subscriptions are generated by nodes. Furthermore, Scribe is not able to generate a well-balanced multicast tree, since MSRN is the parent of all other nodes.

5 Improvements with CrossROAD

In this section we show that using a P2P system optimised for MANETs is highly beneficial to the stability of the Scribe tree. In this set of experiments, we use CrossROAD instead of Pastry, and set the traffic load to 20%, 50% and 100%, respectively. We concentrate on the performance figures related to the quality of the multicast tree, i.e., the average node stress (Figure 10) and the number of re-subscriptions (Figure 11). A complete evaluation of the User Satisfaction parameters, as well as further optimisations of the Scribe algorithm, are subjects of future work.

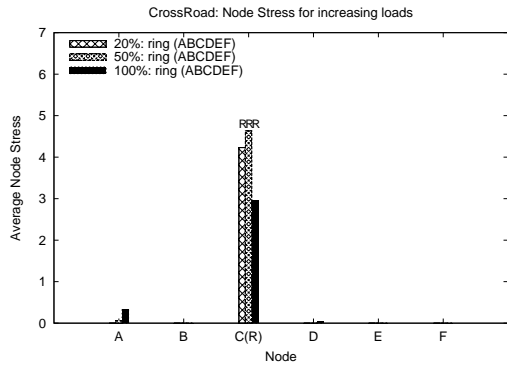


Figure 10. Node Stress with CrossROAD

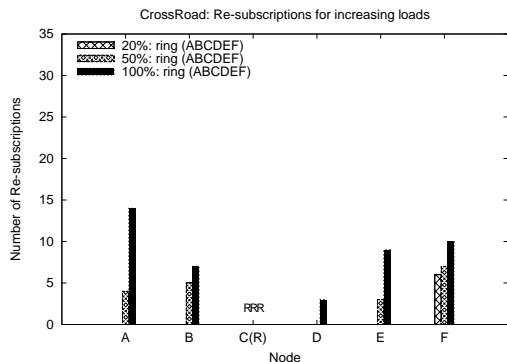


Figure 11. Re-subscriptions with CrossROAD

The first main improvement achieved by using CrossROAD is that neither the overlay network nor the Scribe tree get partitioned. CrossROAD is able to build a *single* overlay network in all the experiments. Furthermore, even at very high traffic loads (e.g., 100%), MSR/N is the *only* root of the Scribe tree. Therefore, CrossROAD is able to overcome all the partition problems experienced when Pastry is used.

Figure 10 clearly shows that the node stress still remains quite unbalanced among the nodes. MSR/N is typically the parent of all other nodes, and this contributes to make it a bottleneck of the system, as highlighted above. This behavior is expected, since it derives from the Scribe algorithm, and cannot be modified by changing P2P system.

Finally, Figure 11 shows that the Scribe tree is more stable (i.e., requires less re-subscriptions) using CrossROAD instead of Pastry. To be fair, we have to compare Figure 11 with both Figures 8 and 9. It is clear that CrossROAD outperforms Pastry when used on top of AODV. The “20%” case of CrossROAD should be compared with the “OLSR 20%” case of Figure 8, since in both experiments the overlay network is made up of all nodes. The num-

ber of re-subscriptions measured at node F is the same in both cases, while it is higher at node E when Pastry is used. The CrossROAD “50%” case shows a higher number of re-subscriptions with respect to the “OLSR 50%” case in Figure 8. However, it should be noted that in the latter case the overlay network encompasses less nodes, and hence the congestion is lower. It should also be noted that, with the same nodes in the overlay network, with the same protocol stack and traffic load, Pastry experiments may suffer MSR/N crashes (Figure 9). In this case, the number of re-subscriptions is much higher than in the CrossROAD case. Finally, results in the CrossROAD “100%” case should be compared with the “OLSR 80%” case of Figure 9, since the overlay network is the same in both experiments. CrossROAD achieves comparable performance, and at some nodes it outperforms Pastry, even if the application traffic is significantly higher.

5.1 Overlay management overhead

In the previous section we have shown that adopting CrossROAD significantly improves the performance of Scribe. In this section we highlight that one of the main reasons for this improvement is the big reduction of the network overhead. This is a key advantage in MANET environments.

Figure 12 shows the network load experienced by nodes A, C and by the two nodes which just act as routers, during the Pastry “OLSR 80%” experiment in which MSR/N crashes⁴. Each point in the plot is computed as the aggregate throughput (in the sending and receiving directions) over the previous 5-seconds time frame. We take into consideration the traffic related to the whole network stack, from the routing up to the application layer. Specifically, nodes A and C are representative for WB nodes, pointing out the difference with nodes that just work as routers. The discrepancy between the curves related to node A and C confirms that the MSR/N node has to handle a far greater amount of traffic with respect to the other WB nodes, due to the Scribe mechanisms. Furthermore, it should be noted that the curves related to the two routers can hardly be distinguished in Figure 12, since they are about 400Bps. This means that the lion’s share of the load on WB nodes is related to Pastry, Scribe and the WB application.

Figure 13 plots the same curves, but related to the “100%” CrossROAD experiment. Also in this case, MSR/N (node C) is more loaded than the other WB nodes. However, by comparing Figures 13 and 12 we can highlight that the Pastry network load is far higher than the CrossROAD network load. By considering the average value over all nodes in the MANET, the Pastry load is about 3 times greater than the CrossROAD load. More specifically, the average load

⁴We do not take into account AODV experiments, since OLSR has clearly shown to outperform AODV.

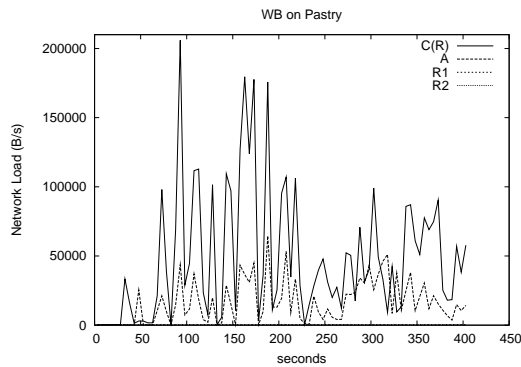


Figure 12. Network Load with Pastry

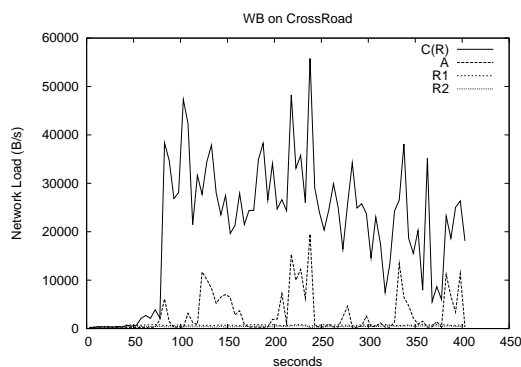


Figure 13. Network Load with CrossROAD

of C and A is 48.5 KB/s and 16.5 KB/s in the Pastry case, while drops to 21.1 KB/s and 2.96 KB/s in the CrossROAD case. The reduction of the network load achieved by CrossROAD is thus 56% at node C and 82% at node A. Since the other stack components are exactly the same, CrossROAD is responsible for this reduction⁵. Furthermore, it should be noted that, during several time intervals, the load of node A is just slightly higher than that of “routing” nodes. This suggests that the additional load of CrossROAD management with respect to the routing protocol is very limited.

6 Conclusions

Results presented in this paper allows us to draw the following conclusions. Pastry and Scribe seem not to be good candidates to support group communication applications in MANET environments. Pastry is particularly weak during the bootstrap phase, causing the overlay network to be partitioned into several subnetworks, and some nodes to be unable to join application services. Further partitions may occur in the Scribe tree due to congestion at the Pastry level. Finally, the delivery algorithm implemented by

⁵The actual reduction is even higher, since the application-level traffic is 100% in the CrossROAD case.

Scribe generates a severe bottleneck in the tree, which is highly prone to get overladed. All these limitations result in unacceptable levels of packet loss and delay for applications. Many of these problems can be avoided by adopting a cross-layer optimised P2P system such as CrossROAD. Thanks to the interactions with a proactive routing protocol CrossROAD is able to avoid all the partition problems experienced with Pastry, and to drastically reduce the network overhead. Clearly, CrossROAD cannot solve the problem of bottlenecks in the Scribe trees. Therefore, optimised version of Scribe are required for group communication applications such as WB to be really developed in MANETs.

References

- [1] 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*, in press.
- [2] M. Castro, P. Druschel, A-M. Kermarrec and A. Rowstron, “SCRIBE: A large-scale and decentralised application-level multicast infrastructure”, *IEEE Journal on Selected Areas in Communication (JSAC)*, Vol. 20, No. 8, October 2002.
- [3] 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”, *Infocom 2003*, San Francisco, CA, April, 2003.
- [4] M. Conti, G. Maselli, G. Turi, and S. Giordano, “Cross layering in mobile ad hoc network design”, *IEEE Computer*, Feb. 2004.
- [5] F. Delmastro, “From Pastry to CrossROAD: Cross-layer Ring Overlay for Ad hoc networks”, in *Proc. of Workshop of Mobile Peer-to-Peer 2005*, in conjunction with the PerCom 2005 conference, Kauai Island, Hawaii, Mar. 2005.
- [6] M. Dischinger, “A flexible and scalable peer-to-peer multicast application using Bamboo”, Report of the University of Cambridge Computer Laboratory, 2004, available at <http://www.cl.cam.ac.uk/Research/SRG/netos/futuregrid/dischinger-report.pdf>.
- [7] P. Gunningberg and H. Lundgren and E. Nordstrom and C. Tschudin, “Lessons from Experimental MANET Research”, *Ad Hoc Networks Journal*, Special Issue on “Ad Hoc Networking for Pervasive Systems”, Vol. 3, Number 2, March 2005.
- [8] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems”, *Middleware 2001*, Germany, November 2001.
- [9] OLSR, Andreas Tonnesen, Institute for informatics at the University of Oslo (Norway), <http://www.olsr.org>.
- [10] AODV, Dept. of Information technology at Uppsala University (Sweden), <http://user.it.uu.se/henrik/aodv/>.
- [11] F. Dabek and B. Zhao and P. Druschel and J. Kubiatoicz and I. Stoica, “Towards a common API for Structured Peer-to-Peer Overlays”, *Proc. of the the 2nd International Workshop on Peer-to-peer Systems (IPTPS’03)*, Berkeley, CA, Feb. 2003.