Router assisted overlay multicast

Konstantinos Katsaros, Nikolaos Bartsotas and George Xylomenos
Mobile Multimedia Laboratory
Department of Informatics
Athens University of Economics and Business
Patision 76, Athens 104 34, Greece
E-mail: ntinos@aueb.gr, nickbar86@hotmail.com and xgeorge@aueb.gr

***Abstract*— While multicasting is considered valuable for content distribution, it is not widely supported on the Internet. Content providers have instead turned to peer assisted content distribution in order to efficiently serve large numbers of clients via unicast, thus removing the bandwidth bottleneck from their side. The redundant unicast transmissions of the same packet are not avoided however, they are just distributed between the peers. Since peer assisted content distribution represents a major fraction of total Internet traffic, a more efficient distribution scheme would be of great interest to users and network operators alike. For this reason, we reconsider overlay multicast as a potential solution for mass content distribution. We present an overlay multicast scheme inspired by Scribe that exploits cooperative access routers so as to improve the multicast content distribution trees produced. We investigate the properties of our scheme compared to both regular Scribe and IP multicast over Internet-like network topologies, via a full fledged simulation platform that can be used as a basis for the realistic evaluation of multicast based content distribution applications.**

I. Introduction

It has been long realized that the Internet is evolving from a network connecting pairs of end hosts to a substrate for information dissemination [1]. Indeed, a major part of today's Internet traffic is due to content distribution applications, in most cases peer assisted applications [2]. Peer assisted, or peer to peer, applications are regarded as very efficient for content distribution purposes, for both the content providers (in terms of required bandwidth) and the end users (in terms of download times). However, in terms of network resources this approach can be very inefficient: many nearby nodes may download the same data from a faraway node instead of from one another, since they make their choices independently [2].

The source of these problems is the Internet's lack of a multicast facility suitable for efficient content distribution. While IP multicast has been available for more than a decade, it has not been widely adopted. One reason for this is that IP multicast routing does not scale well: unlike unicast addresses that can be aggregated in a single routing entry per network area, nearly identical multicast addresses refer to completely different member sets, therefore routers must allocate memory and perform signaling separately for each group [3]. Unfortunately, there are no gains to be made by supporting multicast, unless if all routers support it, therefore there are no clear incentives for individual routers to start providing this service.

The lack of adoption of IP multicast by network providers has led to the development of a variety of alternative end host based solutions, which require no assistance from the network. One option is to employ an *Application Layer Multicast* (ALM) scheme [4], where multicast is simulated by multiple unicast transmissions between group members: the sender transmits packets to some members, those members relay them to others, and so on, until all members are served. In ALM schemes however each group member needs to be aware all other members in order to achieve good routing performance, meaning that these solutions are not scalable.

A more scalable option is to create multicast trees over a *Distributed Hash Table* (DHT) substrate that can route packets based on identifiers, such as Pastry [5]. One such system, Scribe [6], uses the identifier based routing of a DHT substrate to provide a *rendez vous* (RV) point between senders and receivers to a group in a distributed manner. While Scribe seems promising, its performance is known to be quite worse than that of IP multicast [7]; in this paper we show that it is even worse when taking into account the entire path between the source and the receivers. Furthermore, it is hard to simulate realistic applications on top of Scribe so as to determine application level performance, as existing simulators either sacrifice realism for scalability [6], [7] or vice versa [8].

In order to make Scribe more attractive for content distribution, we have modified it to create distribution trees over the topology of the access routers serving the end hosts, thus pruning redundant tree branches and shortening the distribution paths. We evaluate the performance of the proposed scheme against regular Scribe and IP multicast via a full fledged simulation platform over Internet-like network topologies. The driving force behind our work is to support the evolution of the Internet towards a content centric architecture [1]. In the *Publish-Subscribe Internet Routing Paradigm* (PSIRP) project [9] we are working on an Internet architecture based on publish-subscribe principles throughout the protocol stack. To realize this paradigm, we need both applications that operate in a publish-subscribe manner and network mechanisms for rendez vous and data distribution. A choice committed to by the PSIRP project is the reliance on multicast as the main method of data delivery. Therefore, a multicast scheme that can efficiently support the needs of publish/subscribe applications is crucial for the success of the PSIRP project.

The outline of the remainder of this paper is as follows. In Section II we explain the concept of overlay multicast and its router assisted variant. In Section III we describe our simulation environment in terms of network topologies

and scenarios simulated, as well as scalability. In Section IV we present performance results for the trees produced by our router assisted overlay scheme compared to regular Scribe and IP multicast. In Section V we discuss the feasibility of deploying our scheme and the need for a full fledged simulation platform. We summarize our conclusions in Section VI.

II. Overlay multicast

A. Overlay multicast with Scribe

As mentioned above, the absence of IP multicast support has led to the emergence of overlay solutions that do not require network support, such as those based on DHT substrates. In DHT substrates like Pastry [5] a uniform identifier space is distributed among the participating nodes; these are regular end hosts that use the underlying IP transport transparently to the routers. These nodes co-operate to efficiently route data tagged with a specific identifier to the node assigned with that part of the identifier space. To facilitate this process, each DHT node maintains overlay routing state that allows it to relay a received packet to another node whose part of the identifier space is closer to the packet's identifier, until the node actually responsible for that identifier is reached. The advantage of such schemes is that both the amount of routing state required per node and the maximum number of hops required to reach any other node scale logarithmically with the number of nodes, a critical feature in the context of mass content distribution.

On the other hand, packets following overlay routes no longer follow the shortest path towards their destination node. Unlike other DHT schemes (for example, Chord [10]), Pastry attempts to minimize this side-effect, a property that motivates its use in this paper. By employing proximity metrics, such as the number of IP hops or the round trip time, Pastry takes network locality into account: among the possibly many DHT nodes that are closer to a packet's identifier, and which could thus continue relaying the data, Pastry chooses the closest one with respect to the employed proximity metric.

The Scribe [6] system achieves multicast distribution over any DHT substrate, not necessarily Pastry [7], by mapping the name of each group to an identifier and making the node responsible for that identifier the RV point of the group. Receivers join the group by sending a join message towards the group identifier; as the message propagates towards the RV point, reverse path routing state is established until a node already in the tree is found, thus forming a multicast tree rooted at the RV point. A sender simply routes data towards the group identifier, so that the RV point may then propagate it over the established tree. An important characteristic of Scribe is that multicast routing state is maintained in a decentralized fashion: each node in a tree is only aware of its immediate ancestors and descendants. This is a significant scalability advantage over other overlay multicast schemes (for example, Bayeux [11]) as it means that Scribe does not require excessive signaling traffic in order to gather global state information.

The reliance of Scribe on end hosts may however lead to inefficiencies. An end host that is an interior node in some trees will limit the bandwidth available to all those trees to that of its access link. This can be avoided by exploiting the properties of the underlying DHT to create a set of trees such that each node will be an interior node for only one of them [12]. However, this solution is tied to a specific overlay routing scheme (in this case, Pastry). In addition, an end host that is an interior node even for a *single* tree, may still be a bottleneck: as shown in Figure 1(a), data in transit has to enter and exit the RV point (peer *b*) and the other two internal end nodes *c* and *d*, only one of which (peer *c*) is also a receiver, via their access links. If these access links are asymmetric, the tree bandwidth will be limited by the, typically lower, uplink bandwidth.

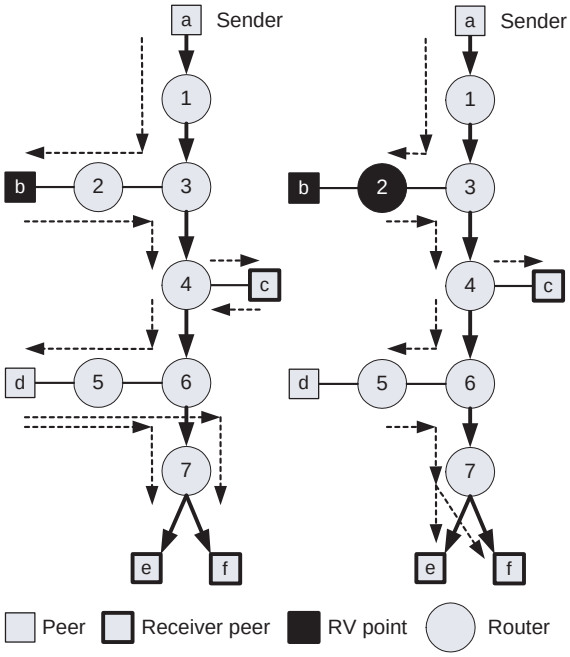

Fig. 1. Overlay multicast: (a) non router assisted (b) router assisted

Another issue is that neighboring end hosts may download the same content via separate tree branches, thus incurring unnecessary network load. For example, in Figure 1(a) the two peers *e* and *f* receive separately the content from their parent in the tree (peer *d*). Note that in this example peers *b* and *d* act as intermediate tree nodes without being receivers. This arises when nodes participating in various multicast groups share the same DHT substrate, so as to amortize DHT maintenance costs among different groups and improve DHT routing performance by increasing the available overlay paths.

B. Router assisted overlay multicast

To avoid these problems, we propose using the access router of a peer as its *proxy* in the DHT substrate *and* overlay multicast scheme. This means that the access router participates in the DHT on behalf of the attached peer. If multiple peers are attached to the same access router, a single place will be held by it in the DHT, that is, the access router will be assigned a single portion of the identifier space, regardless of the number of directly attached peers. An access router will only enter the DHT and act as a proxy if at least one of its attached end hosts is a peer, therefore access routers are not burdened with the signaling overhead of maintaining the DHT unless there is

a reason to do so. In the same manner, the access router also acts as a proxy for the peer in the Scribe trees. This means that the access router is responsible for joining the multicast groups indicated by the attached peers and forwarding the incoming traffic to them. The access router may also participate in a multicast tree as an interior node, subject to its position in the identifier space and the operation of regular Scribe, that is, if in regular Scribe its attached hosts were interior nodes of that tree. In this case, it forwards the incoming traffic to its tree descendants, as well as to any interested attached peers.

The proposed proxy role of access routers presents some significant advantages regarding the characteristics of the created distribution trees. First, as shown in Figure 1(b), data do not need to cross the access links of interior tree nodes at all, only crossing the, typically faster, downlink direction of those access links leading to nodes that are members of the group. For example, data will not cross the access link between peer *d* and router *5* at all, and it will only cross the access link between peer *c* and router *4* in the downlink direction so as to deliver the content to peer *c*. Second, multiple tree branches towards end hosts attached to the same access router can be aggregated in a single branch leading to that access router (in our example router *7*). For the entire distribution tree of Figure 1, router assistance means that a packet transmitted to the group will only cross 12 instead of 20 links with regular Scribe (or 8 with an optimal IP multicast tree), avoiding the uplink direction of access links. Therefore, in router assisted overlay multicast the paths through the distribution trees become shorter and faster, while redundant transmissions over the access links of intermediate nodes are prevented, something especially important in an environment where asymmetric access links are prevalent, as it prevents the (typically slower) uplinks from becoming bottlenecks.

III. Simulation Environment

In order to thoroughly investigate the potential gains of the proposed router assisted overlay multicast scheme, we have performed an extensive set of realistic simulations. One of the main concerns in simulating large scale content distribution is related to the scalability of the simulation. The original performance evaluations of Scribe [6], [7] were focused on scalability so as to investigate the structure of the produced trees and their properties, such as the distribution of the forwarding load in an Internet like environment. In this paper, in addition to investigating the performance of our router assisted overlay multicast scheme with respect to the structure of the produced multicast trees, we also strive to create a simulation environment that can be used for detailed application simulations, being scalable enough to support realistic studies over Internet like network topologies. In our work we used OverSim [8], an overlay network simulation framework for the OMNeT++ simulation environment [13]. The OverSim framework provides implementations of several overlay schemes and applications, among which are the implementations of Pastry and Scribe that our study is based on. The framework also provides a variety of underlying network structures varying from simplistic ones (*SimpleUnderlay*) where no actual routing is performed, to more complicated ones (*IPv4Underlay*) where the complete protocol stack, provided by the INET protocol framework [13], is in operation. The performance results reported in this paper were obtained with OMNeT++ v.3.3, INET v.20061020 and OverSim v.20080416, patched with our modifications, on a system running Ubuntu Linux 7.10 on an Intel Core 2 Duo P9500 processor.

A. Topologies

Our initial concern in building a simulation environment relates to the underlying network topology. OverSim's *SimpleUnderlay* model provides a scalable routing substrate since no network protocols are actually in operation. In this model, packets are directly sent to end hosts by simply using a global routing table, with packet delivery delay being determined by the two communicating ends' distance in the Euclidean space. Furthermore, each end host can be assigned to a logical access network for which the access delay, bandwidth and packet loss characteristics may be set. Though it has been shown that this model provides a scalable solution for the simulation of large numbers of overlay nodes [8], it suffers from serious limitations: the lack of protocol functionality and step-by-step routing are major drawbacks of the model, since important aspects of a real system are neglected, such as the queuing of packets in intermediate nodes (routers) and therefore the packet delays, and even losses, that arise due to network congestion. As a result, this model cannot be used to evaluate the dynamic performance properties of realistic applications.

On the other hand, the *IPv4Underlay* model provides a good approximation of real networking conditions by incorporating the operation of almost all widely deployed networking protocols. We have therefore focused on this model for our work, exploring its memory and processing time requirements for the simulation of very large network topologies. Apart from scalability, the other major drawback of this model is that it lacks support for routing policy weights, such as those produced by the Georgia Tech Internet Topology Model (GT-ITM) [14] that has been used for previous studies of Scribe performance [6], [7]. Instead, the model employs an unweighted shortest path algorithm (Dijkstra's) which calculates the shortest paths between any pair of network nodes, regardless of their placement on the network. Hence, in contrast with reality, it is possible for a path between two routers in a single stub network to pass through several transit routers, something very likely to influence the results produced, especially when it comes to routing issues. By not taking routing policy weights into account, routing paths may become shorter than in reality, and since Pastry employs proximity metrics in the selection of overlay neighbors, this could result in the selection of the wrong node as an overlay neighbor.

In order to avoid routing inaccuracies, we constructed a conversion tool that allows the use of GT-ITM topologies within the OverSim platform. Our tool was implemented as an extension to the BRITE topology generator export tool [13] which already allows the parsing of GT-ITM topologies and the conversion of BRITE topologies into OMNeT++ format. However, the existing tool did not provide any support for

TABLE I
NETWORK TOPOLOGY PARAMETERS

	Topo0	Topo1	Topo2	Topo3
Transit domains	5	7	9	10
Avg. routers per transit domain	5	4	4	5
Stub domains per transit router	5	7	9	10
Avg. routers per stub domain	5	7	9	10
Stub routers	625	1372	2916	5000
Transit routers	25	28	36	50
Total routers	650	1400	2952	5050

weighted topologies, nor did it make a distinction between transit and stub routers, producing flat topologies. We solved these problems by piggybacking the routing weights inside the channel definition of the produced OMNeT++ topology, using fields whose values are not provided by the GT-ITM model, but are instead later read from configuration files. Furthermore, we incorporated the distinction between *transit* and *stub* routers in the tool and translated it into *IPv4Underlay*'s distinction between *backbone* and *access* routers. The support for routing policy weights was completed by employing a weighted shortest path algorithm, leading to a platform that captures all the intricacies of GT-ITM and is therefore comparable to earlier simulation studies based on it.

B. Simulation scenarios

In our simulations we considered a wide range of network topologies ranging from 650 routers to 5050 routers in total. Five different topologies were generated for each network size and all presented results express the average values over those instances. Table I shows the GT-ITM parameters used to generate the topologies [14]. In all topologies, the default link establishment probabilities were used, that is, a link between two transit routers was established with a probability equal to 0.6 and a link between two stub routers was established with a probability equal to 0.42. The target number of end hosts were then randomly attached to the stub routers.

In order to stress the limits of our simulation environment, we first investigated the memory requirements for loading each topology in the simulator, and then tried to strike a balance between the memory requirements of the larger topologies and the limitations they imposed on the number of end hosts participating in the experiments. Most of the results in this paper refer to Topo1 with a variable number of end hosts in the network. Specifically, we simulated three different scenarios with 500 (Sparse), 1000 (Medium) and 4000 (Dense) end hosts, respectively, so as to explore the impact of host density to the performance of the suggested overlay multicast scheme.

Regarding multicast groups and their sizes, a Zipf-like distribution was used for the size of each group, that is, the r-th group had a size equal to $\lfloor Nr^{-1.25} + 0.5 \rfloor$, where N was the total number of overlay nodes, as in [6]. The first group included all overlay nodes, while the last group had a size equal to 11 nodes, a size typical of instant messaging applications [6]. Based on this lower bound and the number of participating hosts, the number of groups in the Sparse scenario was 21, in the Medium scenario it was 36 and in the Dense scenario it was 110. The members of each group were randomly selected from the entire end host population, meaning that each end host may have participated in many groups. For each group, a random identifier was chosen and a *non-member* end host was randomly selected as the sender.

In all scenarios, end hosts first join the overlay (Pastry) network; in our scheme, this means that proxy routers join the overlay network on behalf of their attached end hosts. After the initialization of the overlay network has completed, participating nodes start joining, and therefore forming, the multicast trees. When all nodes have joined the respective trees, the senders send a data packet towards the RV point of each tree to inspect the path between the sender and the RV point. Note that in this paper, unlike in earlier ones [6], [7], we explicitly take into account not only the tree formed between the RV point and the receivers, but also the path from the sender to the RV point, since application performance is determined by the entire path between sender and receivers.

C. Scalability

As shown in [8], OverSim enables the simulation of scenarios with even 100.000 overlay nodes. However, in the simulations presented in that paper, the underlying network was either too simplistic (*SimpleUnderlay*) or too small (*IPv4Underlay* with 20 backbone and 20 access routers). If the same number of overlay end hosts is used with both models, then the memory requirements of the *IPv4Underlay* model will be larger due to additional network elements (routers and links) and their operation. It is also expected that the memory footprint of routers will differ from that of overlay end hosts.

The simulator memory requirements for the considerably larger network sizes described in Table I are presented in Figure 2: the x-axis indicates the size of the network in terms of the total number of participating routers, while the curves indicate the memory footprint of scenarios either without any end hosts or with 1000 end hosts in the proposed router assisted (*Proxies*) or in the regular Scribe scheme (*No Proxies*). It is clear that the memory footprint of the networking topology increases dramatically with the number of participating routers, in a non-linear fashion. This increase is due to the increasing number of links between the participating routers. The memory requirements of the network topologies have a severe impact in the feasibility of large scale scenarios, since, for example, a topology with 5000 access routers and 50 backbone routers, already requires approximately 1800 MB of memory; for a realistic simulation, we also need to add end hosts, along with their access links.

It is important to point out why our proposed router assisted scheme requires less memory than the regular Scribe scheme: as all overlay functionality is provided by the access routers, we did not create the actual end hosts at all, so as to reduce the memory footprint of the simulation. Furthermore, our scheme requires fewer messages for the establishment of the overlay, since each router joins the overlay only once, regardless of the number of end hosts that it is a proxy for. Note also that these messages travel smaller distances since they do not need to

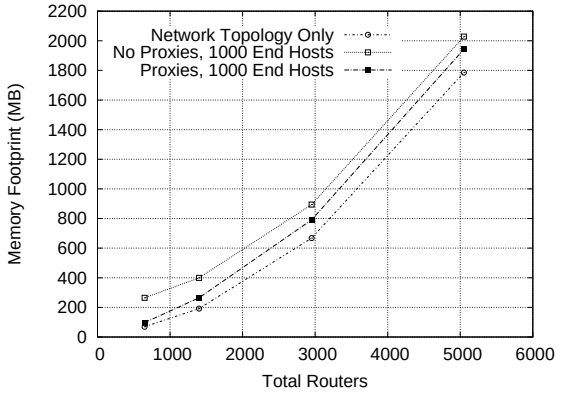

Fig. 2. Simulation memory requirements

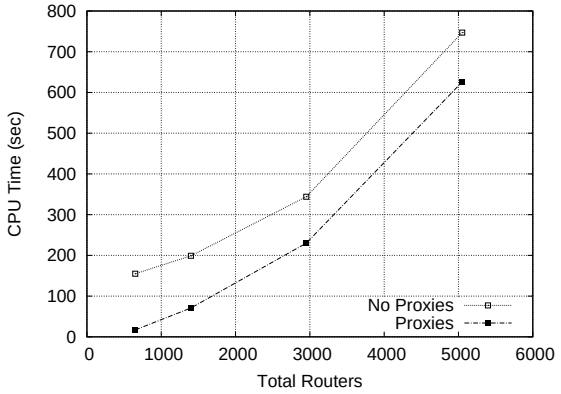

Fig. 3. Simulation processing time

cross the access links; only a single message is required for an end host to initially ask its access router to be its proxy.

Figure 3 shows the processing (CPU) time for the simulation of each scenario, which includes constructing a network topology with 1000 end hosts, running Pastry to establish a DHT substrate, running Scribe to establish multicast trees and sending a message to the RV point of each multicast tree. Again, the resource requirements of the *Proxies* scenario are lower than those of the *No Proxies* scenario. This can be attributed to the lower number of running simulation modules as well as to the avoidance of multiple overlay maintenance messages when many hosts are attached to the same router.

IV. Performance Evaluation

As the proposed scheme alters the structure of the multicast trees produced, we examine below the performance of each scheme in terms of the properties of the resulting trees.

A. Path stretch

The most obvious metric for evaluating overlay routing schemes is the overhead incurred by not strictly following the shortest path IP routes. This is usually expressed by the term *stretch* which is defined as the ratio between the overlay path length (or delay) and the length (or delay) achieved by IP routing. In essence, *stretch* expresses the degree to which the performance achieved is worse than the performance of

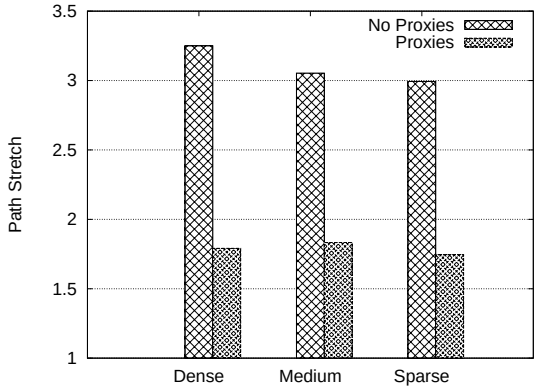

Fig. 4. Path stretch

the underlying IP substrate, a penalty paid in return for the scalability of the overlay solution. Multicast complicates this metric, not only because we need to take into account the entire distribution tree, but also because the Internet does not provide multicast routing in the first place. The usual convention, also followed here, is to assume that IP multicast would use the tree formed by merging the optimal unicast paths between the sender and each receiver. In the absence of multicast, a sender desiring to reach all receivers would have to send duplicate copies of each packet over all those paths.

We define *path stretch* as the ratio between the number of IP hops comprising the path from the sender to a receiver in the multicast overlay tree to the number of hops that comprise the shortest unicast path between these two nodes, averaged over all trees and paths. The overlay paths are comprised of two segments, from sender to RV point and from RV point to receiver. For the first segment we use the shortest path between these two nodes: after the initial message sent to locate the RV point for a group, the sender can cache its IP address so as to avoid the overlay path for subsequent data transmissions. For the second segment we use the path constructed by the overlay scheme. Figure 4 shows the path stretch achieved for the Sparse, Medium and Dense scenarios in Topo1; by definition the stretch of IP multicast is 1. It is clear that our approach yields significantly lower stretch values in all scenarios (40-45% decrease), regardless of end host density.

It should be pointed out that our results diverge from those shown in the papers evaluating Scribe [6], [7] in two ways. First, we measure the *entire* path between the sender and each receiver, as opposed to the path between the RV point and each receiver, since it is the entire path length that applications have to cope with. Second, we use network *hops* rather than delays to calculate stretch, since in a static environment we can only calculate the propagation delay. Since this delay may only represent a small part of the actual delays faced by applications in a dynamic environment where queueing delays due to congestion may be dominant, we believe that defining stretch using these delays would be misleading.

B. Transmission Stretch

Another important performance metric is the efficiency of the multicast trees produced in terms of the total load imposed

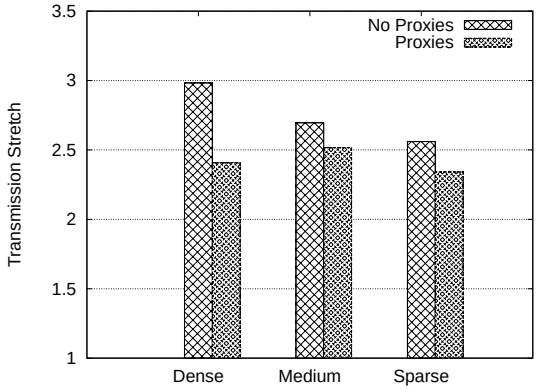

Fig. 5. Transmission stretch

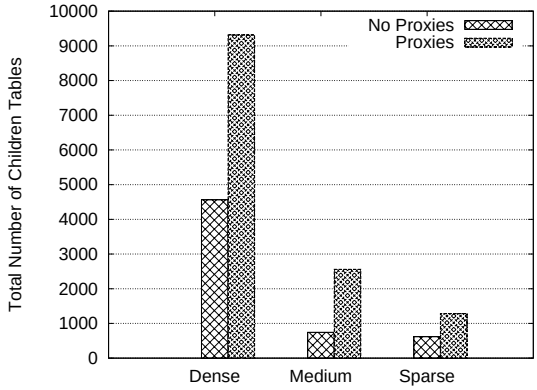

Fig. 6. Node stress: children tables

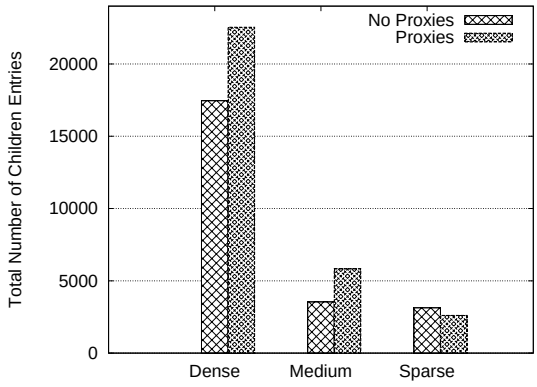

Fig. 7. Node stress: children entries

on the network for data distribution. We define as the *total transmission load* the number of hop-by-hop transmissions required in order for a single packet originating from the sender of each group to reach all group members of the corresponding group. In order to provide a normalized metric, we define *transmission stretch* as the ratio between the total transmission load imposed by each overlay multicast scheme to the total transmission load imposed by IP multicast.

Figure 5 shows the transmission stretch achieved for the Sparse, Medium and Dense scenarios in Topo1; by definition the stretch of IP multicast is again 1. Exploiting router assistance results in a decrease of the transmitted packets in all cases (7-19% decrease). This is due to the fact that the proposed router assisted multicast overlay scheme leads to the formation of multicast trees with fewer IP hops compared to the regular Scribe trees, leading to a more efficient utilization of the network. Recall also that, as mentioned above, our router assisted scheme eliminates packet transmissions in the uplink direction of access links leading to intermediate nodes, thus removing a bottleneck imposed by asymmetric access links.

C. Node Stress

A critical metric of a multicast scheme's performance is the forwarding load it imposes on participating nodes. In a multicast scheme forwarding load can be expressed by the branching factor of each node, that is, the number of descendants it forwards traffic to; in a multi-tree scenario, attention must also be paid to the number of groups served by a node. In Scribe each node maintains forwarding information in a separate *children table* per group, with each table's entries referring to the node's children for that group. Hence, we calculate two metrics of node stress, as in [6], [7]. The first metric is the *number of children tables* maintained by each node, which corresponds to the number of multicast groups the node is forwarding traffic for. In our router assisted scheme, this includes the groups for which a router is not forwarding data to other overlay nodes, but is only acting as a proxy on behalf of one or more attached receivers. The second metric is the *number of children entries* maintained by each node, which corresponds to the number of nodes it is forwarding traffic to, across all multicast groups. Again, in our scheme this includes the group recipients attached to an access router, for which the access router acts as a proxy.

Figures 6 and 7 depict the two node stress metrics for the Sparse, Medium and Dense scenarios in Topo1, aggregated over all nodes in the network; results from other topologies indicate that both metrics are mostly dependent on end host density. The proposed router assisted scheme incurs a significant forwarding state overhead increase, especially considering that this state is concentrated on the access routers serving end hosts participating in the overlay, rather than being distributed among the participating end hosts as in regular Scribe. This increase however is largely an artifact of our definition of the node stress metrics, as both metrics include the proxy entries in the access routers, which do not require the same maintenance overhead as regular Scribe entries.

In order to assess how the amount of forwarding state affects the nodes participating in the overlay multicast routing process, Figures 8 and 9 show the number of overlay nodes per tree, as a function of the number of end hosts participating in the DHT substrate, either directly or via their access routers; for clarity, the figures show only the ten most popular groups in each scenario. As expected, with more end hosts (Dense) and fewer access routers (Topo1), Figure 8 shows that the forwarding load in the router assisted approach is concentrated on fewer nodes than in regular Scribe, while with fewer end hosts (Sparse) and more access routers (Topo3), Figure 9

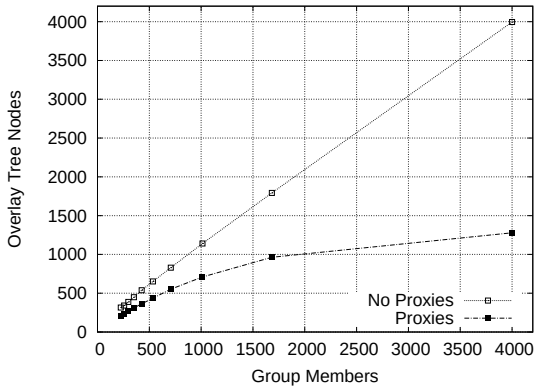

Fig. 8. Number of overlay nodes (Topo1, Dense)

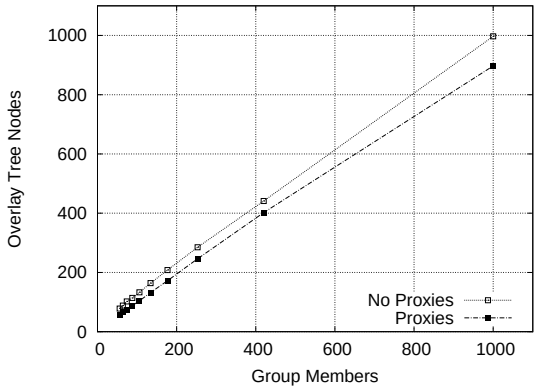

Fig. 9. Number of overlay nodes (Topo3, Sparse)

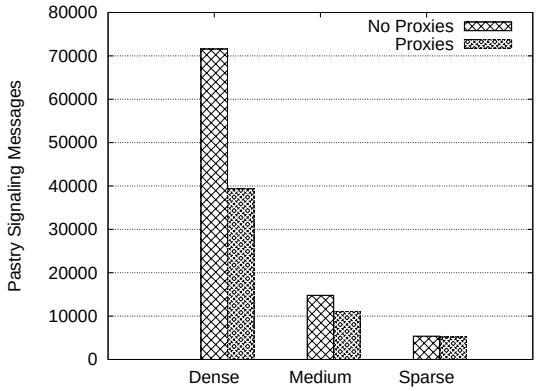

Fig. 10. Pastry signaling overhead

shows that the forwarding load of the router assisted approach is distributed in roughly the same manner as in regular Scribe.

On the other hand, this reduction in the number of overlay participants also reduces the number of overlay signaling messages exchanged to establish the DHT substrate (Pastry). As demonstrated in Figure 10 for the Sparse, Medium and Dense scenarios in Topo1, the router assisted approach greatly reduces (by 45%) the signaling overhead for DHT maintenance in the Dense scenario, as in this case the access routers participate in the overlay on behalf of many end hosts. On the other hand, in the Sparse scenario each router hardly ever serves more than one end host, thus we see only a slight reduction (4%); in the Medium scenario the reduction lies between these extremes (25%). Note that in our router assisted scheme these messages travel shorter distances since they do not need to cross the access links to and from the end hosts. Therefore, there exists a tradeoff between placing the burden of forwarding and tree maintenance on the access routers, and reducing path stretch, link stress and DHT related signaling.

V. Discussion

After comparing the performance of regular Scribe against its router assisted variant, one is likely to come up with two questions: first, how feasible is it to expect access routers to participate in such a scheme and, second, how the full fledged simulations in this paper differ from simulations in the literature. Regarding the first question, since overlay multicast is a response to the lack of support for IP multicast, proposing that access routers should maintain an overlay multicast routing scheme on top of a DHT substrate seems counter intuitive. However, while in IP multicast *all* routers must participate in multicast routing, otherwise multicast is useless, in our scheme access routers can *optionally* take part in DHT maintenance and overlay multicast routing: the DHT substrate and the overlay multicast scheme can operate over end-hosts only, albeit with reduced efficiency, as shown. Either way, overlay multicast does not suffer from the routing state scalability problems that plagued IP multicast.

Furthermore, while routers have no incentives to start supporting IP multicast, in our scheme individual access routers have clear incentives to act as proxies, even if no other access routers do so. First, they reduce their traffic load, since they eliminate transmissions over both their access links (for routers *2* and *5* in Figure 1) and their router-to-router links (for routers *5* and *7* in Figure 1). Second, they provide an enhanced service to their customers, as the end hosts will experience lower latencies and higher bandwidths in multicast applications. These are the same arguments that motivated network providers to offer Web proxy services to their clients. Since routers do not provide higher layer functionality, an alternative would be to deploy dedicated proxies, co-located with the access routers, in order to support overlay multicast (i.e. transport layer, Pastry and Scribe functionality). The results presented above are still valid for this alternative deployment scenario, assuming negligible communication costs between the access routers and their co-located application layer multicast proxies.

Regarding the second question, a key observation is that routing is only a means to an end, which in our case is improving the performance of content distribution applications via multicast. The static tree properties provided by the custom overlay simulators described in the literature are insufficient to characterize the dynamic performance of actual applications, and this is exactly why we have evaluated the feasibility of executing full fledged simulations of overlay multicast: only by including a model of the application in the simulation will we be able to predict application level performance.

As a first example of the impact of application behavior, consider the tradeoffs discussed above. If an application uses

multicast to distribute very large amounts of data to many recipients, our router assisted approach will be preferable to regular Scribe due to its reduced path stretch and link stress. If however an application uses multicast to send small amounts of data to many small groups, regular Scribe may be preferable due to its reliance on end hosts only. In both cases, only a full fledged simulation can assess metrics such as path delays, as these depend on the load placed on each link.

As a second example, consider the case of applications with dynamic multicast group membership. In this paper as, to the best of our knowledge, in all previous studies of Scribe, it is assumed that the DHT is fully formed before Scribe signaling begins and never changes thereafter. A full fledged simulation incorporating dynamic group membership would reveal that node arrivals and departures, in addition to increasing the maintenance requirements of the underlying DHT, also affect the multicast trees themselves. In our router assisted scheme where access routers participate in the overlay on behalf of multiple hosts, group dynamics may have a smaller effect than in regular Scribe, depending on the application under study.

As a third example, consider the idea of global access router participation to the DHT substrate, regardless of the presence of interested end hosts. This would lead to a very stable DHT substrate since routers are less volatile than end hosts, as well as to improved routing due to the additional possibilities offered, but it would also cause the DHT maintenance overhead to increase due to the larger number of nodes participating in the overlay. Again, in order to assess whether this change would be beneficial to application level performance or not, one would need to undertake a full fledged simulation study incorporating actual application behavior.

VI. Conclusions

In this paper we have presented and evaluated an overlay multicast scheme based on Scribe, whereby the overlay network is composed solely of the access routers to which the participating end hosts are attached. Our target in this paper was twofold. First, we wanted to establish a realistic simulation environment and explore its potential scalability limitations. As part of this effort, we developed a tool for the utilization of GT-ITM topologies in the OverSim platform and studied the limitations imposed by the size of the underlying network topologies. Second, we wanted to demonstrate the gains related to the exploitation of co-operative access routers. Our results show a significant improvement in the resulting tree properties such as path stretch and link stress, as well as a reduction in the overlay network establishment and maintenance overhead, at the cost of increased forwarding overhead for the access routers. In addition, the data presented in this paper indicate that the simulation platform and our overlay multicast routing approach can be used to study the dynamic properties of multicast based content distribution applications. This is especially important for the publish-subscribe applications that will arise in the context of the network architecture designed by the PSIRP project.

Acknowledgements

The work reported in this paper was supported by the ICT PSIRP project under contract ICT-2007-216173.

References

[1] V. Jacobson, *If a Clean Slate is the solution, what is the problem?* http://cleanslate.stanford.edu/seminars/jacobson.pdf.
[2] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, “Should internet service providers fear peer-assisted content distribution?,” in *Proc. of the Internet Measurement Conference*, pp. 63–76, 2005.
[3] B. Zhang and H. Mouftah, “Forwarding state scalability for multicast provisioning in IP networks,” *IEEE Communications*, vol. 41, no. 6, pp. 46–51, 2003.
[4] Y.-H. Chu, S. 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.
[5] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems,” in *Proc. of the Middleware Conference*, pp. 329–350, 2001.
[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, “SCRIBE: A large-scale and decentralized application-level multicast infrastructure,” *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 100–110, 2002.
[7] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, “An evaluation of scalable application-level multicast using peer-to-peer overlays,” in *Proc. of the IEEE INFOCOM*, vol. 2, pp. 1510–1520, 2003.
[8] I. Baumgart, B. Heep, and S. Krause, “OverSim: A flexible overlay network simulation framework,” in *Proc. of the IEEE Global Internet Symposium*, pp. 79–84, 2007.
[9] PSIRP Project Team, *PSIRP Project Home Page*. http://www.psirp.org.
[10] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *Proc. of the ACM SIGCOMM*, pp. 149–160, 2001.
[11] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz, “Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination,” in *Proc. of the 11th NOSSDAV*, pp. 11–20, 2001.
[12] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “SplitStream: High-bandwidth multicast in cooperative environments,” in *Proc. of the Symposium on Operating Systems Principles*, pp. 298–313, 2003.
[13] A. Varga, *OMNeT++ Simulator Home Page*. http://www.omnetpp.org.
[14] E. Zegura, K. Calvert, and S. Bhattacharjee, “How to model an internetwork,” in *Proc. of the IEEE INFOCOM*, vol. 2, pp. 594–602, 1996.