Multicast State Distribution by Joins Using Multiple Shortest Paths

Damien Magoni, Pascal Lorenz

Multicast State Distribution by Joins Using Multiple Shortest Paths

Damien Magoni
Université Louis Pasteur – LSIIT
67400 Illkirch, France
magoni@dpt-info.u-strasbg.fr

Pascal Lorenz
Université de Haute Alsace – GRTC
68008 Colmar, France
pascal.lorenz@uha.fr

Abstract

The lack of resources in routers will become a crucial issue with the deployment of state storing protocols. In particular, single or any source multicast protocols will most probably take over large amounts of resources for maintaining multicast tree information. The aim of this paper is to study the possibility and benefit of using multiple shortest paths in order for a new member to reach a multicast tree. Such a mechanism would not reduce the overall amount of state information in the network but it would distribute this amount more evenly among all routers. The idea is to use alternate shortest paths provided by the underlying unicast routing protocol to avoid saturated routers, that is, routers that can not or do not want to store any more multicast state information. As the simulation results are very sensitive to the topology, we have used subgraphs of an Internet map. We have then simulated our multipath join mechanism and have found that depending on the tree size, the use of our mechanism can increase successful join attempts by up to 55% when the network is half saturated.

Keywords

Internet, join, multicast, multipath, state.

I. Introduction

State storing protocols, such as Intserv model QoS protocols or any-source multicast protocols, do raise the concern of scalability inside the network because they need to maintain information in intermediate network points such as routers that were not initially designed to fulfill such tasks. On the other hand, Chalmers *et al.* have shown in [1] that multicast outperforms unicast with just a few receivers when considering bandwidth usage. Between 20 and 40 receivers, multicast provides a least 55% increased efficiency over unicast, 75% with 150 receivers and up to 90% for groups with 1000 receivers. Thus, it seems worthwhile to support multicast in the network despite the cost incurred in state storage. That is why the research community is still trying to deploy Internet-wide multicast protocols while simultaneously looking to reduce as much as possible the amount of states stored in the network or to optimize the location of those states. Our solution helps optimizing the location of multicast states by placing them on alternate shortest paths. Our paper contains two main sections. In section III we describe the basic principle of our solution and provide hints on how to implement it. Then in section IV we present the performance results obtained by simulation for assessing the significance of our solution.

II. Related work

Despite the claim of Holbrook *et al.* that a router could handle for instance a million of active single source multicast channels [2], the concern about the multicast forwarding state scalability for any or single source multicast has been growing among the network research community. This issue has been investigated at the inter-domain level by Wong *et al.* in [3]. They have found that the increase in peering among backbone

networks has led to more multicast forwarding state at a handful of core domains but less state in the rest of the domains. They also have found that the scalability of multicast state with respect to session density follows a power law (i.e., the fraction of routers that are stateful grows proportional to a constant power of multicast group size). Their findings have been reinforced by Chalmers *et al.* in [1] who have found that most inter-domain multicast trees actually do share a common shape at both the router and AS levels. In other words, they have discovered that the range of possible shapes of inter-domain multicast trees is constrained by the underlying network connectivity. Several solutions have already been proposed to solve the multicast state scalability problem. Tian *et al.* have defined a dynamic tunnelling multicast in [4] to avoid storing multicast information in non-branching forwarding routers. Thaler *et al.* have proposed a multicast forwarding state aggregation scheme in [5] to reduce multicast routing entries, while Zappala *et al.* have investigated alternate path routing techniques for multicast in [6]. More recently, Jun-Hong Cui *et al.* have proposed several schemes aimed at the aggregation of multicast states in [7], Zhang *et al.* have defined a forwarding state reduction method for delay-constrained multicasting in [8] and Yang *et al.* have proposed an optimized allocation scheme based on explicit multicast in [9]. Finally Gorinsky *et al.* have studied in [10] the impact of attacks by inflated subscription in multicast congestion control, further emphasizing the fact that creating many multicast states can also lead to denial of service attacks. However none of them is related to our approach of alternative joins and resulting state distribution.

Our paper proposes a new solution, that can be combined with the previous ones, and was designed:

- to increase the chances for the join message of a new member to reach its desired multicast tree by avoiding blocking routers,
- to distribute the states among the routers by storing them in routers accepting more multicast states.

We insist on the fact that our solution does not reduce the amount of multicast states but it spreads these states over more routers thus balancing the state repartition. Our method can be used by network layer tree construction and maintenance sparse mode multicast protocols such as [2], [11]. It's worth noticing that any protocol using a tree structure for data distribution and a source oriented unicast joining mechanism, and used in the Internet, can be concerned by the results of our study.

III. USING MULTIPATH FOR JOINING MULTICAST TREES

In this section we describe the assumptions that we make in order for our solution to work. We also look at the multicast state storing capabilities of popular routers and further define what is a saturated router. We then describe the how our solution is functioning and we give hints for its implementation.

A. Assumptions

First we assume that we are in the Internet, that hosts are using the TCP/IP protocol suite and that packets are forwarded by routers. We suppose that QoS and multicast and any other state storing protocols are in use and are consuming memory in the routers. A saturated router is a router which currently refuses to store any additional multicast state entries whatever the reason.

We suppose that a new member is willing to join a multicast group and thus its corresponding multicast tree. In our experiments, we build the multicast tree as defined in the Protocol-Independent Multicast - Single Source Multicast (PIM-SSM) draft, because it is the multicast protocol that will mostly be deployed in the near future. The creation of the multicast tree happens as follows: we randomly select a source, then we add members (i.e., receivers) one by one and we store a state in every router crossed along the unicast path from each receiver towards the source if it does not already have one (i.e., if it is not yet belonging to the tree). Thus the tree obtained is a shortest path tree rooted at the source. When we build a tree, we do not consider the saturated routers, we suppose that the tree has been created without problem. Its only when

TABLE I. ROUTER STATE STORING CAPABILITIES.

Vendor	Series	Route engine	Max. memory	Max. nb of multicast states
Cisco	12000	PRP-2	4GBytes	256K
Juniper	E	SRP-40G+	2GBytes	128K

we simulate the join message of a new receiver going towards the already created tree that we take the saturated routers into account. In the case of an Any-Source Multicast (ASM) protocol such as PIM-SM, the join message would be targeted towards a Rendez-vous Point (RP) instead of the multicast source. Also in this latter case the tree would be a shortest path tree rooted at the RP. We have not investigated the impact of these differences in our preliminary simulations and we leave this for future work. However the results of Chalmers *et al.* encourage us to think that the two kinds of trees would still have similar topologies. The session density is equal to the number of receivers in the tree expressed as a percentage of the total number of routers. Thus the tree size is related to the session density but not equal to it (i.e., non-receivers internal tree nodes must also be counted).

Finally we assume that the underlying unicast routing protocol is giving us multipath information. That is, for a given destination, the routing system is able to indicate to us all the next hop routers that lead to this destination by a shortest path. The shortest paths do not need to be disjoint (i.e., the paths can have common parts). Intra-domain protocols such as OSPF, IGRP and IS-IS can provide such multipath information whereas RIP can not.

B. Router saturation

It is very hard to obtain any precise information concerning the state storing capability of the routers made by major hardware companies. Table I shows some information collected on the vendors websites for the best hardware configuration possible as of early 2004. Cisco claims that a 12810 router can hold 256000 multicast groups however it does not tell if this is valid for the default SDRAM capacity or for the maximum SDRAM capacity. Similarly Juniper specifies that an E-series router can store 16384 multicast groups but does not precise the necessary amount of memory. If we suppose that it is stored in a SRP-5G card containing 256MB of SDRAM, we conclude that storing a multicast group in a Cisco or a Juniper router requires an average of 16K bytes. This seems a lot but an implementation of PIMSM in a router typically requires three tables: a Multicast Routing Information Base for managing multicast shortest paths needed for Reverse Path Forwarding, a Tree Information Base (TIB) for the management of all the trees crossing this router and a Multicast Forwarding Information Base derived from the TIB for efficiently forwarding multicast packets. Moreover the entry sizes in the last two tables depend on the number of interfaces involved. All these data structures can consume a lot of memory. Thus the current maximum number of multicast states that can be stored by today high end routers is an order of magnitude of 10^5. Needless to say that most of the routers will have a capacity of one or two order of magnitude lower.

On the other side, the class D contains roughly less than 228 (i.e., more than 268M) potentially usable multicast addresses which can be used by ASM protocols and with the channels defined in PIM-SSM, the number of group addresses can grow up to the number of hosts multiplied by 224 (i.e., more than 16M) channels per host. Thus if 1% of the end hosts in the Internet, which today means around 1 to 3 million hosts, can use multicast protocols and particularly PIM-SSM, and if every host creates 5 SSM channels, this will generate 5M to 15M different entries in the crossed routers. The core routers of the Internet will be crossed by a large part of these trees as shown by Wong *et al.* in [3] and thus will easily have to store more than 50% of all the possible states. This means at least 2.5M entries to store for these core routers, much more than the high end routers of Table I can cope with. Thus router saturation can clearly become a

frequent phenomenon if multicast is widely deployed and accessible to customers. If this issue is not talked about now, it is mainly because most operators and providers do not offer multicast services to customers. We define a saturated router as a router which can not (or does not want to) store any more multicast state entry at a given moment. Of course this saturation state varies in time because multicast states are dynamically created and deleted. In our experiments, we adopt a macroscopic view and assume that x % of the routers are saturated at a given moment (snapshot). Furthermore, saturated routers are selected in a purely random fashion. We are aware that if we pile a huge amount of multicast trees, following the results of Wong *et al.* [3], a few routers will pile a state for nearly each tree (i.e., especially routers close to the network core) while many others will only pile a few states. However, these heavily loaded routers will most probably be well equipped in hardware by their operators while the others farther from the core may be less equipped and thus prone to saturate faster. Thus we believe that it is satisfactory to select them randomly for the moment.

C. Basic principle and implementation hints

When the new member wants to join, it sends a join message along the unicast path towards the source as shown in figure 1 and as defined in the PIM-SSM draft. The join message contains an index defining the output interface that will be used (among all the possible multipath interfaces leading to the multicast tree source) and that we call the multipath index. This multipath index is updated in every router crossed, thus it is meaningful only for the last routed successfully crossed.

In the figure, the new member is 4 hops away from the source. The default unicast routing path goes to the left, however the second router is saturated, it can not or refuses to store a state (it is not in this specific tree). Our mechanism implies that the saturated router should send a message back to the previous router in order to tell it to try an alternate path. The reject message contains the address of the multicast tree source and the unmodified multipath index (i.e., which keeps exactly the same value as in the join message). When the previous router (i.e., the ok node on the figure) gets the reject message, it should update the multipath index to the next output interface that will be used and send the join message to this next interface leading to the source by another shortest path.

As already said, in order to be able to do this, the underlying unicast routing protocol must provide the multipath information. Thus it must know all the interfaces that lead to a given router by different paths having the same length. Intra-domain routing protocols such as OSPF, IGRP and IS-IS can handle the management of multipath information.

The router uses the address of the tree source as well as the multipath index (i.e., the index of the last used output interface) in order to determine the next possible multipath interface to the tree source. It then sends a new join message to the next possible neighbor located on a shortest path towards the source (with an updated multipath index). In our example, upon reception of the reject message, the join message is sent to the right router (i.e., the in-tree 2 node on the figure) and the join succeeds (as this router is already in the tree). The join attempt stops here. An interesting side effect of our solution is that it creates a state in a router that has some storage space left. By filling up the memory of non-saturated routers, our mechanism provides a better use of the network resources by distributing states in routers that may have been left under-used otherwise. If at a given router, all attempts to join the tree source by anyone of the possible multipath interfaces fail, then the process stops. No backtrack takes place, as this would require to store more information in the packet (i.e., the indices of all crossed routers) and a more complex processing. In this case, the join fails as it does without our mechanism when a join message encounters its first saturated router on its way to the tree source. Kindly notice that in the figure, there is one saturated router for 10 routers, thus the saturation density is 10% and the tree contains 2 receivers, thus the session density is 20%. As all the possible alternate paths are always shortest paths (such as the default path), the routing is

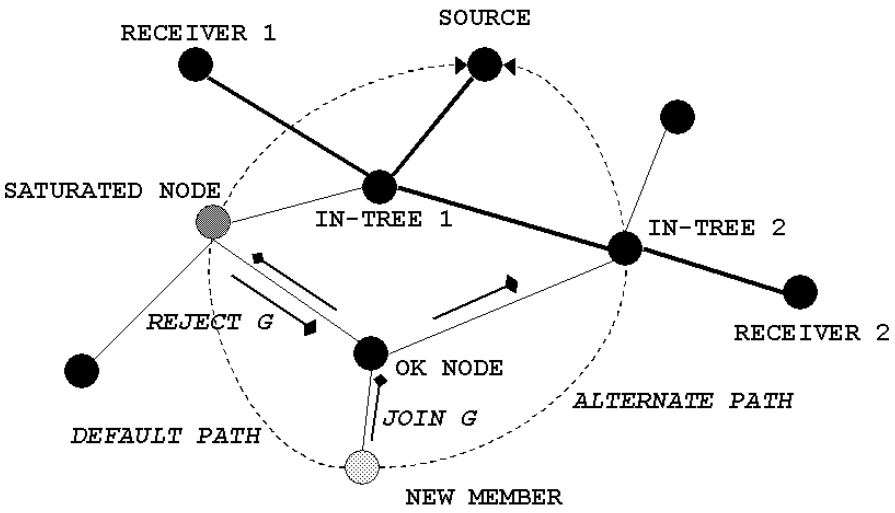

Fig. 1. Example of a new member using multipath to reach the tree.

always optimal and so is the alternate path used to connect to the tree. Also if the unicast routing tables are coherent (i.e., no temporary loops due to routing problems), our mechanism does not create loops either.

To be efficient and feasible, our mechanism should be as simple as possible. We do not create a new protocol as we can implement our mechanism in any existing multicast tree management protocol. We have seen before that we need:

- to store the multipath index (i.e., an index defining the last used multipath interface) in the join message,
- to define a reject message containing the address of the multicast tree source and the multipath index,
- an unicast routing protocol supporting multipath.

We have chosen to use only multiple shortest paths unlike Zappala *et al.* who use non shortest paths (e.g. shortest path plus one hop, plus two hops, etc.) for alternate path routing techniques for multicast in [6]. In fact, it is much more complicated to obtain the non shortest path information and it is not managed by the unicast routing protocol, which means that we should handle it ourselves. We prefer to use only the information given by the unicast routing protocol to keep our mechanism practical. The delivery of the messages of our mechanism is best-effort, thus increased reliability can be provided by sending several messages before stopping. The pseudo-code for our mechanism consists in two functions and is listed on Figure 2.

In a real implementation such as in PIM-SM for example, we only need to:

- modify the Join/Prune message in order to store the index of the last used output interface (it could be stored in the 9th 8-bit reserved field or in a newly added 32-bit field if 256 interface possibilities are not enough),
- define a Reject message that contains the index of the last used interface (the same index as above) and the address of the multicast tree source.

The key point is that our mechanism does not need any modification to the IP forwarding mechanism

```
Upon reception of a JOIN packet:
  1. If router is saturated
  2.   Send REJECT message to previous router by using the
       reception interface if known or the receiver address
  3. Else
  4.   Create a multicast state for this group/channel
  5.   Determine the output interface that leads to the
       source and set the MULTIPATH INDEX accordingly
  6.   Send a JOIN message with this MULTIPATH INDEX
       by this interface

Upon reception of a REJECT packet:
  1. Find the next usable multipath output interface by
     using the MULTIPATH INDEX in the packet
  2. If all interfaces leading to the source have been tried
  3.   Stop the JOIN procedure
  4. Else
  5.   Update the MULTIPATH INDEX in the packet to the next
       output interface that leads to the source and
       set the MULTIPATH INDEX accordingly
  6.   Send a JOIN message with this MULTIPATH INDEX
       by this interface
```

Fig. 2. Alternate shortest path algorithms.

and more importantly it does not need to store any state information in the router, apart from those already stored by the multicast protocol. Thus it is scalable with respect to the memory state of the routers.

IV. Experiments

In this section we present results obtained by simulation for evaluating the efficiency of our proposal. There are currently no known technique that provides the same service as our mechanism. Thus we can not compare our solution to other approaches excepted the default multicast tree join approach. We show that the ability to use other shortest paths instead of the default unicast path if it is blocked can bring a substantial gain concerning the success of the join messages.

A. Settings

Ensuring the accuracy of the topologies used for our simulations on multipath and multicast is crucial. These topologies must be similar to a real Internet map with regard to their topological properties, otherwise the simulation results could be biased by the shape of the underlying topology as we have shown in [12]. We have used a map gathered by our *nec* software cartographer [13] as an input for our simulations. We have used our map sampling method upon our 47k-node router-level *nec* map in order to produce Internet-like networks containing 4000 routers. This is a typical size of a large core Internet ISP including its customers and peers as shown in [14]. These networks do comply with the degree power laws found in the Internet by Faloutsos *et al.* in [15] and especially the power laws on the number of multiple distinct shortest paths found in our previous work [16] which directly relate to the amount of redundancy links in the Internet.

TABLE II. SIMULATION PARAMETERS.

Parameter	Values
Network size	4000 routers
Saturation density	5, 10, 15, 20, ..., 80
Session density	2, 4, 6, 8, ..., 40

TABLE III. MEASURED METRICS.

Metric
Number of multiple shortest paths from new member to tree
Number of valid shortest paths
Number of blocked shortest paths
Is default shortest path valid?

In our experiments, we adopt a macroscopic view and assume that x % of the routers are saturated at a given moment (snapshot) as explained in section III-B. We have analyzed various network states by using saturation densities ranging from 5% to 80% and various multicast tree sizes defined by their session densities (i.e., the percentages of network nodes that are receivers in the multicast tree) and ranging from 2% to 40%. Table II shows the parameter space of our simulations.

As the process of generating multicast trees and selecting new members involves random selection (and thus random rolls), we have used a sequential scenario of simulation [17] to produce the results shown in the next section. We have used the Mersenne Twister code [18] for producing the random numbers needed in our simulation. As the random rolls are the only source of randomness in our simulation, we can reasonably assume that the simulation output data obey the central limit theorem. We have performed a terminating simulation where each run consists in picking a new member and determining the number of multiple shortest paths between it and the source, stopping at the multicast tree, and counting how many of these paths are blocked by a saturated router (i.e., one run is the time horizon). Thus in our simulations, we have measured all the metrics shown in Table III. Ideally we should have created a multicast tree, selected a new member, measured the metrics and carried out a sequential checkpoint. However this would have been too costly in terms of computing power. Thus we have performed a sequential checkpoint each time after having created a tree and selected and measured the metrics for 500 new members. All the simulation results have been obtained assuming a confidence level of 0.95 with a relative statistical error threshold of 5% for all measured metrics.

B. Results

In order to have an idea of the number of multiple shortest paths between a new member and its tree, we plot the average (of all the runs) of this number in figure 3 as a function of the session density. This number is quite high, being always between 2.2 and 8.2. When the session density increases and thus the tree size increases, the number of multiple paths decreases. Indeed when the tree size grows, the distance between the new member and the tree decreases and so does the possibility of having alternate shortest paths. Furthermore this plot has the typical shape of a heavy tailed distribution. We plot this distribution on a log-log scale in figure 4. We can see that a linear regression performed on the data holds a correlation coefficient of 0.99 which accounts for the presence of a power law in this distribution. This comes at no surprise as many topological properties of Internet-like graphs do comply with power laws. Concerning this plot, it is most probably influenced by the power laws on the number of multiple distinct shortest paths found in our previous work and explained in [16]. These plots already indicate that there will be a lot of

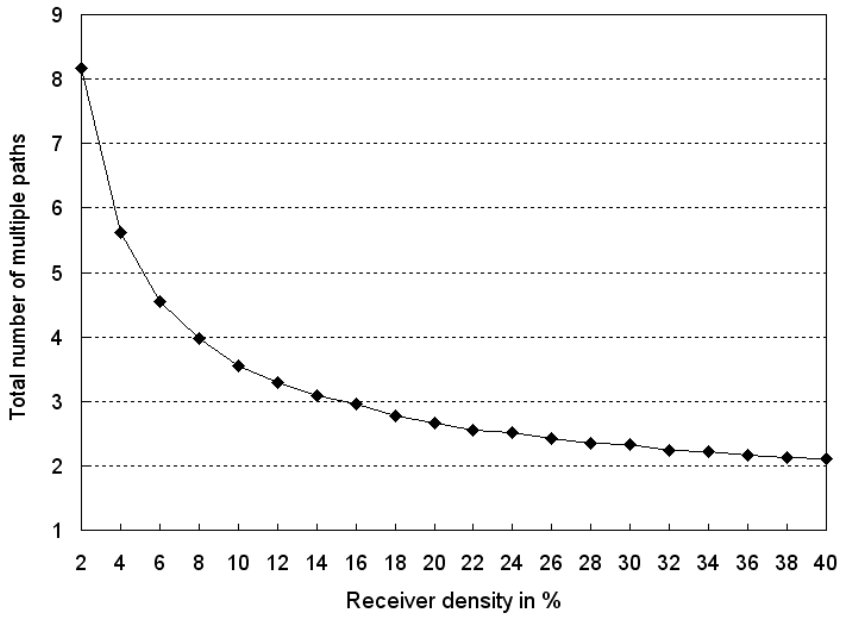

Fig. 3. Mean total number of multiple paths for a new member joining a tree as a function of the density of the receivers in the network.

alternate shortest paths available.

In figure 5 we plot the percentage of joins that have succeeded *by using the default unicast path* (with respect to all joins, successful or not) as a function of the saturation density. These can be viewed as the results obtained by using the regular join method defined in any or single source multicast protocols. That is, without using our mechanism. We can see that this percentage is very dependent on the saturation density. When the receiver density is high (i.e., large trees), the relationship is nearly linear, whereas when the receiver density is low (i.e., small trees), the relationship tends to be non-linear. That is, the percentage of success decreases faster in the low saturation ranges and slower in the high saturation ranges. To explain this phenomenon, we can see that when the network is mid saturated (i.e., between 30% and 70%) and the tree is large, the new member will be closer to it and thus will have a lesser chance of being blocked by a saturated router. This difference is no longer significant when the network is highly saturated in which case most of the default joins fail or when the network is not saturated in which case most of the default joins succeed. To see the influence of the session density, we plot the percentage of joins that have succeeded by using the default path as a function of the session density in figure 6. We can see that the percentage of success does not depend much on the session density and thus the size of the tree. The percentage seems to be influenced only when the densities are low (i.e., under 8%). This plot is interesting as it shows that the chances of joining a popular multicast tree do not really depend on its size but much more on the saturation state of the network.

In figure 7 we plot the number of joins that have failed with respect to the number of all possible joins (expressed as a percentage), as a function of the saturation density. Depending on being unsuccessful or not, the default unicast path is or is not counted in the failed joins. We can see that this percentage is very dependent on the saturation density. When the receiver density is high, the relationship is nearly linear, whereas when the receiver density is low, the relationship tends to be non-linear. That is, the percentage of blocked paths increases faster in the low saturation ranges and slower in the high saturation ranges. This plot shows that, most probably because the alternate paths are scare in most parts of an Internet-like

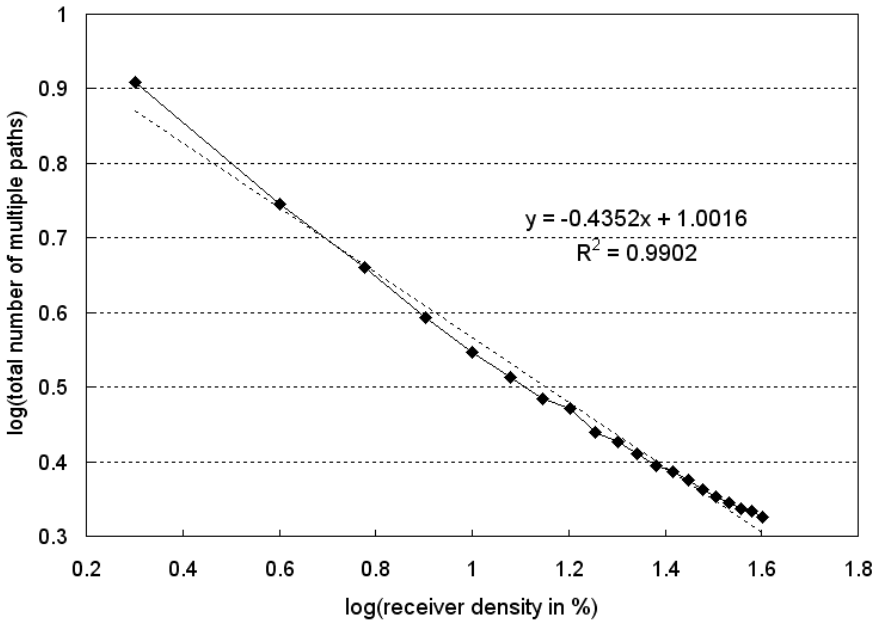

Fig. 4. Logarithm of the mean total number of multiple paths for a new member joining a tree as a function of the logarithm of the density of the receivers in the network.

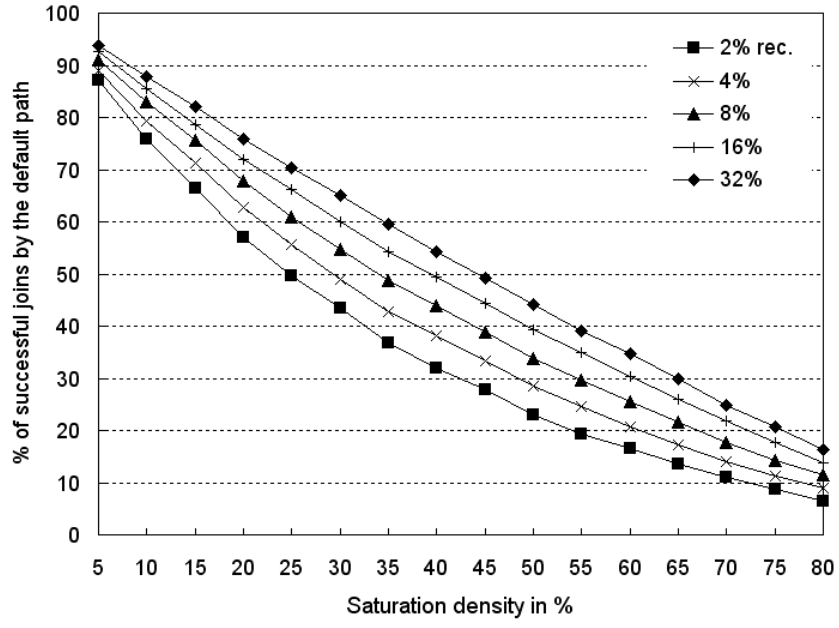

Fig. 5. Percentage of successful joins obtained by using the default path as a function of the density of the saturated routers in the network.

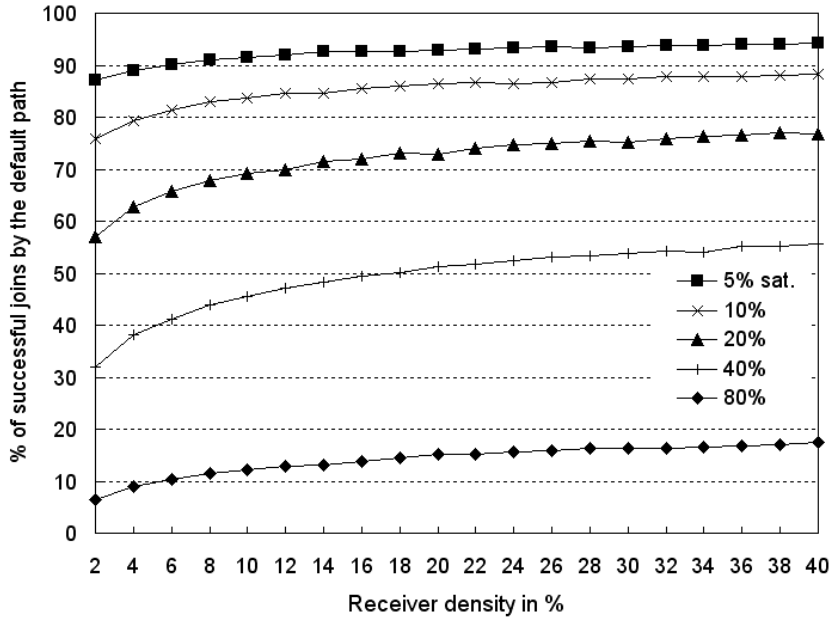

Fig. 6. Percentage of successful joins obtained by using the default path as a function of the density of the receivers in the network.

topology, the join attempts will become impracticable above a given saturation level as most of the possible paths will be blocked. As before, in order to see the influence of the session density, we plot the percentage of failed joins as a function of the session density in figure 8. The percentage of blocked paths slowly decreases when the session density increases. The decrease is stronger when the session densities are low (i.e., under 10%). This plot shows that the chances for joining a large multicast tree are slightly better than a small one at a given saturation state.

The last two plots show the benefit in percentage that can be obtained by using our alternate shortest path join mechanism instead of using only the default path. Figure 9 shows the percentage of additional successful joins obtained by using a non-blocked alternate path, when the default unicast path is blocked by a saturated router, as a function of the saturation density. We can see that the gain is at least linear with respect to the saturation density and when the tree size is small it falls back a little bit when the network is saturated above 50%. These results are very promising and encourage us for implementing our mechanism. Figure 10 shows the percentage of additional successful joins (defined as in the previous plot) as a function of the receiver density. We can see that the gain decreases when the receiver density increases, especially in the low density range. We think, as above, that when the trees are small, the chances of the new member being far away from the tree are higher. Thus the number of shortest paths between the joining node and the tree increases, providing more alternate paths to turn around saturated routers. It's worth noticing that the wavy aspect of some parts of the these plots are probably an artefact of our simulation. Our technique consisting in running a large number (i.e., 500) of runs before performing a sequential checkpoint may slightly affect the statistical results. If we had chosen smaller values for the number of runs we may have obtained smoother plots. Also we have used a relative statistical error threshold of 5% which may not be accurate enough to avoid this phenomenon.

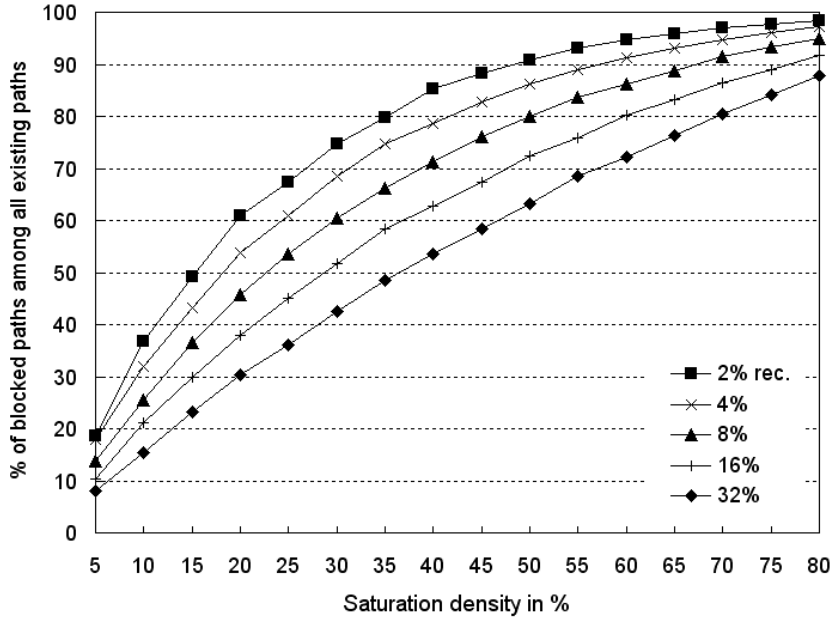

Fig. 7. Percentage of blocked paths among all multiple paths for a new member joining a tree as a function of the density of the saturated routers in the network.

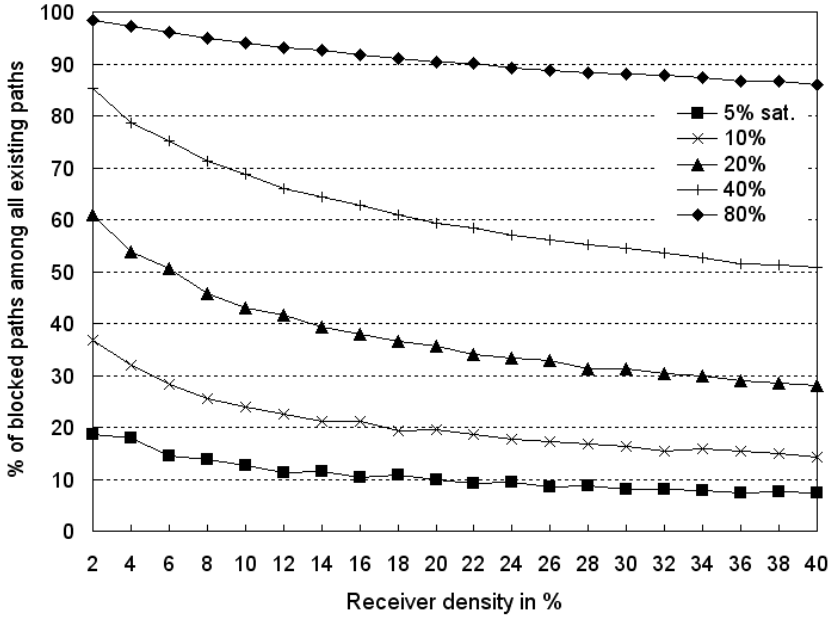

Fig. 8. Percentage of blocked paths among all multiple paths for a new member joining a tree as a function of the density of the receivers in the network.

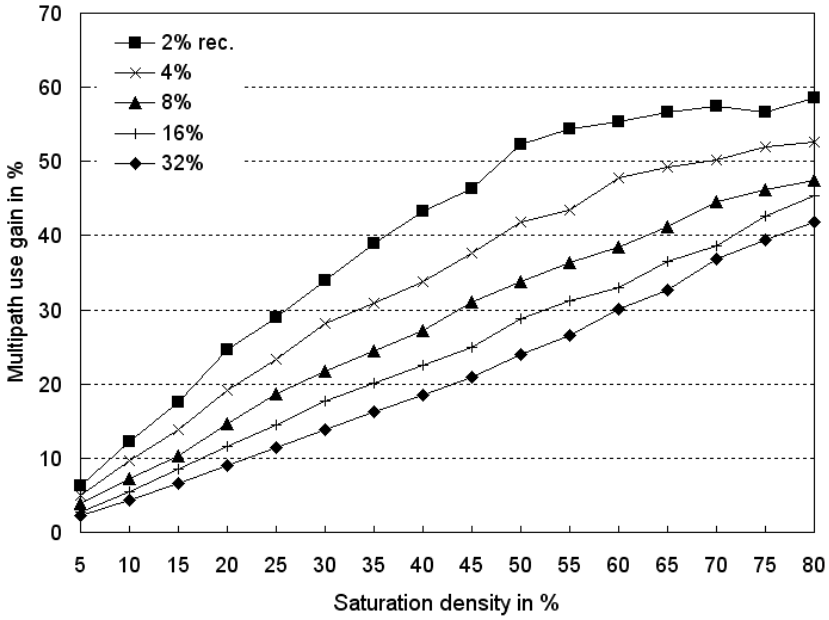

Fig. 9. Percentage of additional successful joins (i.e., gain) obtained by using any non-blocked non-default path as a function of the density of the saturated routers in the network.

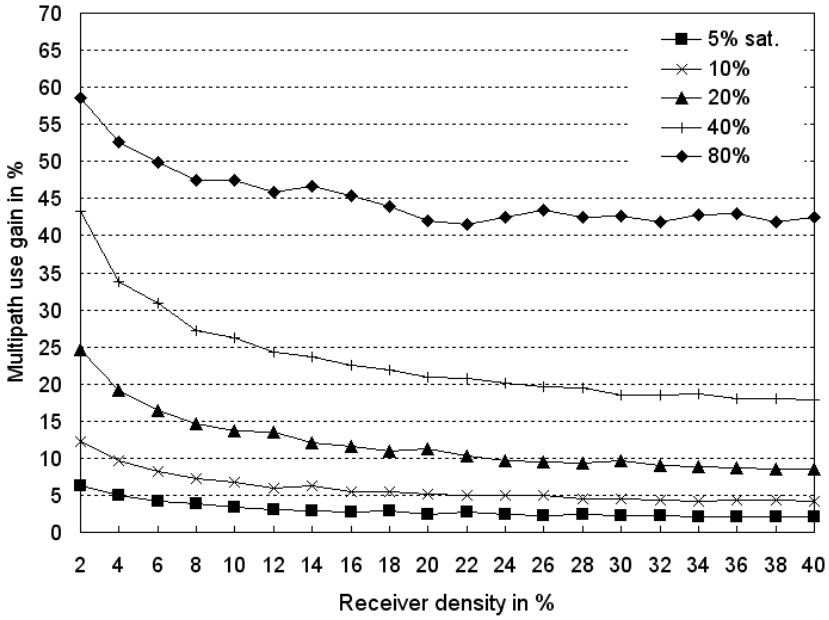

Fig. 10. Percentage of additional successful joins (i.e., gain) obtained by using any non-blocked non-default path as a function of the density of the receivers in the network.

V. Conclusion

Improving the efficiency of multicast protocols is important to promote their deployment in the Internet. In this paper we have proposed a very simple mechanism designed to take benefit of multiple shortest paths in order to increase the chances for a new member to join a multicast tree while at the same time distributing the multicast states among non saturated routers. To carry out realistic simulations, we have used an accurate router level map of the public Internet core created in our previous work. We have used our map to model large intra-domain topologies. We have found that when such networks are heavily saturated (i.e., more than 50%), the increase of successful alternate joins over default joins ranges from 25% to 55% depending on the session density. Thus it seems very profitable to implement our mechanism in routers, especially given the fact that it needs only minor modifications to multicast tree management protocols (such as PIM-SSM) and it does not store any state information in the routers. We are however aware that the support of multipath is not systematically available in intra-domain and that our solution is not currently applicable in inter-domain as BGP does not handle multipath information.

References

[1] R. Chalmers and K. Almeroth, "On the topology of multicast trees," *IEEE/ACM Transactions on Networking*, 2002.

[2] H. Holbrook and D. Cheriton, "Ip multicast channels: Express support for large-scale single-source applications," in *Proceedings of ACM SIGCOMM'99*, 1999, pp. 65–78.

[3] T. Wong and R. Katz, "An analysis of multicast forwarding state scalability," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 419–427, 2000.

[4] J. Tian and G. Neufeld, "Forwarding state reduction for sparse mode multicast communication," in *Proceedings of IEEE INFOCOM'98*, 1998, pp. 711–719.

[5] D. Thaler and M. Handley, "On the aggregatability of multicast forwarding state," in *Proceedings of IEEE INFOCOM'00*, 2000, pp. 1654–1663.

[6] D. Zappala, "Alternate path routing for multicast," in *Proceedings of IEEE INFOCOM'00*, 2000, pp. 1576–1585.

[7] J.-H. Cui, D. Maggiorini, J. Kim, K. Boussetta, and M. Gerla, "A protocol to improve the state scalability of source specific multicast," in *Proceedings of the IEEE GLOBECOM'02*, November 2002, pp. 1910–1915.

[8] B. Zhang and H. Mouftah, "Forwarding state reduction for delay-constrained multicasting in ip networks," in *IEEE Globecom*, 2003.

[9] D.-N. Yang and W. Liao, "Optimizing state allocation for multicast communications," in *IEEE Infocom*, 2004.

[10] S. Gorinsky, S. Jain, H. Vin, and Y. Zhang, "Robustness to inflated subscription in multicast congestion control," in *ACM SIGCOMM*, 2003.

[11] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C.-G. Liu, P. Sharma, and L. Wei, "Protocol independent multicast-sparse mode (pim-sm) : Protocol specification," Internet Engineering Task Force, Request For Comments 2362, June 1998.

[12] D. Magoni and J.-J. Pansiot, "Influence of network topology on protocol simulation," in *Proceedings of the 1st International Conference on Networking*, Colmar, France, July 2001, pp. 762–770.

[13] M. Hoerdt and D. Magoni, "Completeness of the internet core topology collected by a fast mapping software," in *Proceedings of the 11th International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, October 2003, pp. 257–261.

[14] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *Proceedings of ACM SIGCOMM'02*, Pittsburgh, PA, USA, August 2002.

[15] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of ACM SIGCOMM'99*, Cambridge, Massachusetts, USA, August 1999, pp. 251–262.

[16] D. Magoni and J.-J. Pansiot, "Analysis of the autonomous system network topology," *ACM Computer Communication Review*, vol. 31, no. 3, pp. 26–37, July 2001.

[17] A. Law and W. Kelton, *Simulation Modelling and Analysis*, 3rd ed. McGraw-Hill, 2000.

[18] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.