# Automatic Link Numbering and Source Routed Multicast

Visa Holopainen, Raimo Kantola, Taneli Taira, Olli-Pekka Lamminen

▶ **To cite this version:**

Visa Holopainen, Raimo Kantola, Taneli Taira, Olli-Pekka Lamminen. Automatic Link Numbering and Source Routed Multicast. 4th International Conference on Autonomous Infrastructure, Management and Security (AIMS), Jun 2010, Zurich, Switzerland. pp.123-134, 10.1007/978-3-642-13986-4_19 . hal-01056637

**HAL Id: hal-01056637**

**https://inria.hal.science/hal-01056637v1**

Submitted on 20 Aug 2014

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

# Automatic Link Numbering and Source Routed Multicast

Visa Holopainen, Raimo Kantola, Taneli Taira, and Olli-Pekka Lamminen

Aalto University, Department of Communications and Networking, P.O. Box 3000, 02015 TKK, Finland
{visa.holopainen,raimo.kantola,taneli.taira,olli-pekka.lamminen}@tkk.fi

**Abstract.** We present a new paradigm for multicasting. Our paradigm combines two ideas: a simple distributed algorithm for automatically numbering network links with a small amount of numbers, and a novel forwarding method. We describe our paradigm in detail and discuss its benefits and drawbacks qualitatively and quantitatively.

**Key words:** multicast, link numbering, Steiner trees

## 1 Introduction

Internet Protocol (IP) based multicast does not scale, because it requires core routers to keep a forwarding table entry corresponding to every multicast group (in the worst case). As the number of multicast groups grows with the upheaval of IP television and publish/subscribe applications, this is no longer feasible.

Another approach for multicasting called LIPSIN (Line Speed Publish/ Subscribe Inter-Networking) was presented in [1]. The headers of LIPSIN carry a bloom filter that describes every link of the multicast tree. While being an interesting concept, LIPSIN has serious drawbacks, *e.g.*: **DoS**; LIPSIN prevents loops by dropping packets that came in from a wrong interface with respect to the packet's bloom filter. However, since two different trees can have the same bloom filter representation, this causes Denial of Service (without virtual links, which dramatically increase the protocol complexity and sometimes the amount of state). **Forwarding to wrong links**; By design, LIPSIN forwards traffic to some links that are not a part of the intended multicast tree. While it is possible to reduce the amount of such "false positives" by certain optimizations, they are an intrinsic property of bloom filters. If false positives are unacceptable, some other multicast mechanism is needed. **No local recovery**; In IP multicast all link/node failures are repaired locally. LIPSIN does not specify this, and also it cannot recover from single node/link failures using bypass tunnels (without virtual links). The reason is that normally LIPSIN disallows forwarding packets to the interface from which they came in, which is needed in many recovery scenarios. **Large overhead**; By design, bloom filters are relatively long bit strings - in LIPSIN 256 bits is suggested. While this may be acceptable in large networks providing IP television service, small networks may find this wasteful.

Traditional (IP/MPLS) source routing suffers from high header overhead as well as from the lack of a protocol mechanism for multicasting. Also, source routes in packet headers reveal network topology to eavesdroppers [1]. Hence source routing is not used in current networks even for unicasting. We propose a Source Routed Multicast (SRM) paradigm that aims to rectify the drawbacks traditionally associated to source routing.

### 1.1 Overview of SRM and qualitative comparison to LIPSIN, IP multicast, and traditional source routing

The main differences of SRM compared to traditional source routing are the following: First, to make the header overhead in SRM as small as possible, we present *a simple distributed algorithm that automatically numbers core network links with a small amount of numbers*, leading to a small number of bits needed in packet headers (Section 2). We find empirically that this mechanism keeps header overhead very low. Second, we introduce *a protocol mechanism for multicasting with source routes*. More specifically, we propose to include additional bits into the packet header's "link stack" that indicate if a given core router needs to forward the packet to its customer interface(s) (Section 3). We find empirically that this mechanism does not introduce prohibitively high cost in terms of link usage (Section 4). Third, to give some protection against topology leaking, we use a *random arbitrary-length padding in the "address stack" of packet headers* (Section 3). We circulate the padding in the header so that there is no way (at least in principle) for an eavesdropper to know the location of padding in the header, and hence no way to separate real link numbers from the padding [1].

The high-level overview of SRM is illustrated in Fig. 1. SRM is intended to be used within a single Autonomous System (AS), and it is (implicitly) a Layer 2 protocol. However, it might be possible to extend it to a multi-AS setting by allowing ASes to advertise source routes (in some obfuscated format to hide topology) across AS boundaries. However, since arguably (due to bandwidth use based billing) there seems to be very little incentive for ASes to provide transit service for multicast, we will only focus on the intra-AS scenario.

The main philosophical drawback of SRM compared to LIPSIN is that in SRM the *access* routers need to keep state (except in broadcast access networks, *e.g.* wireless). However, we claim that maintaining state in access routers is acceptable for two reasons: First, the amount of state needed in access routers is typically small compared to the amount of state potentially needed in core routers, and second, since access network topology is typically a tree or a ring, the access routers do not need complicated routing protocols. We claim that SRM provides smaller state for *core* routers than LIPSIN, because LIPSIN needs to cache certain packets (for details see [1]).

[1] The third aspect of our design is presented solely to point out that it may be possible to hide topology with source routing. Hence (and due to lack of space) we will only mention a few (extremely heuristic) mechanisms in Section 2 and in Section 3 that should be associated to the use of padding in order to reduce the risk of topology leaking. This is a potential topic for future work.

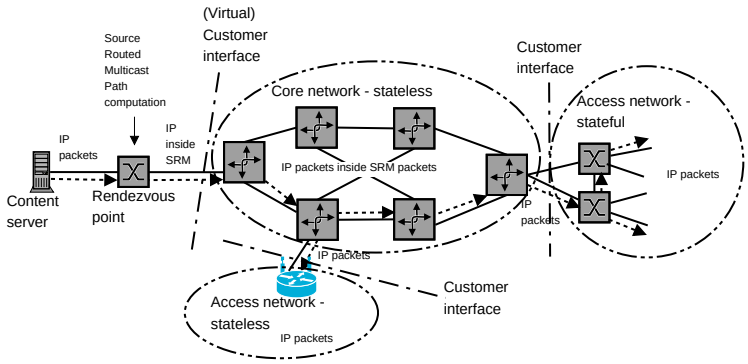



**Fig. 1.** SRM architecture overview. Note that SRM introduces more packets to the interface between the rendezvous point and its adjacent core router (also known as "stress" [15]). Hence one should implement this interface virtually instead of physically.

SRM facilitates local recovery from single link/node failures, just like MPLS [9][2]. Also, SRM forwarding does not involve any cache lookups, and hence (at least in principle) it should always be fast.

In LIPSIN a core router needs to AND the header of each incoming packet with each of its interface addresses. Hence it is slow if routers are not equipped with interface processor. On the other hand, in SRM the amount of per packet logic operations is constant.

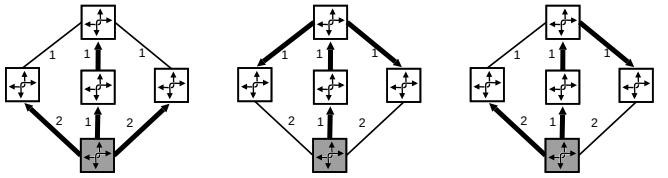
In IP networks a random path is selected among equal cost paths between two nodes. Hence by using IP we cannot always measure the performance of a given path. Also, LIPSIN does not facilitate this (without node IDs in packets). This is a concequence of the fact that in LIPSIN there may be several end-nodes even if we try to pin a certain path (because of forwarding to wrong links). On the other hand, SRM facilitates much easier active measurements. This point should become clear once we describe our paradigm in detail in Section 3.

An example illustration of the differences in the cost-optimal multicast trees facilitated by the three approaches is presented in Fig. 2. There the gray router is the core router next to the multicast source, and the white routers are next to multicast destinations. In the figure LIPSIN provides the lowest cost while IP multicast yields the highest cost. In Section 4 we find that this is the case in general.

### 1.2 Related work

SRM utilizes source-routing of multicast trees. Previous papers [4, 5] have studied source-routing of general multicast trees, in which branching is allowed outside

[2] In the case of local recovery the core routers need to be capable of adding the "label stack" corresponding to the bypass tunnel into the packet header.



(a) IP multicast - naïve [10] trees are used for multicasting (cost=6).

(b) LIPSIN - cost-optimal Steiner arborescences may be used (cost=4).

(c) SRM - cost-optimal set of paths may be used (cost=5).

**Fig. 2.** Example of the lowest-cost multicast trees provided by different approaches. Numbers represent link costs.

the source node. However, source routing of general trees is very complicated in terms of forwarding. We avoid this complexity by (intuitively) disallowing branching outside the source. Many papers, *e.g.* [6, 16], have addressed the problem of computing multicast trees with degree constraints. Our approach is different in the sense that we do not impose a degree constraint on the source node, while for other nodes the (out)degree constraint is 1 (intuitively). Many papers, *e.g.* [15], have studied MPLS-based multicasting, in which branching is allowed only in certain routers. Our approach differs from this line of work, since we utilize explicit source routing instead of (IP/MPLS) routing tables. Many papers, *e.g.* [2], have addressed vehicle routing problems, which capture the essence of our problem. These papers typically focus on computation, while we will focus more architectural and protocol aspects.

## 2 Automatic link numbering algorithm of SRM

The first question one might ask is, why should we number links when they already have MAC-addresses. Could we not use them for source routing? The problem is simply overhead. For example, if we would source-route the path in Fig. 3 with 48-bit addresses, the resulting address stack would be 392 bits long at the first node, while with SRM it is only 33 bits long (see Fig. 4).

When thinking about automatic numbering, the first thing that might come into mind would be to number the nodes (core routers) of the network with network-wide unique numbers. This would facilitate hop-by-hop forwarding with the help of routing and forwarding tables, as well as tunneling multicast packets by listing only the destination node numbers in the packet header. However, this approach runs into trouble if we connect two large networks that have already numbered nodes. Also, it would not facilitate removing routing tables from core routers. Hence our automatic link numbering algorithm assigns a number for each *link* of the network instead. These numbers are *not* unique within the network. Instead, the aim is to use the smallest amount of numbers (and hence the

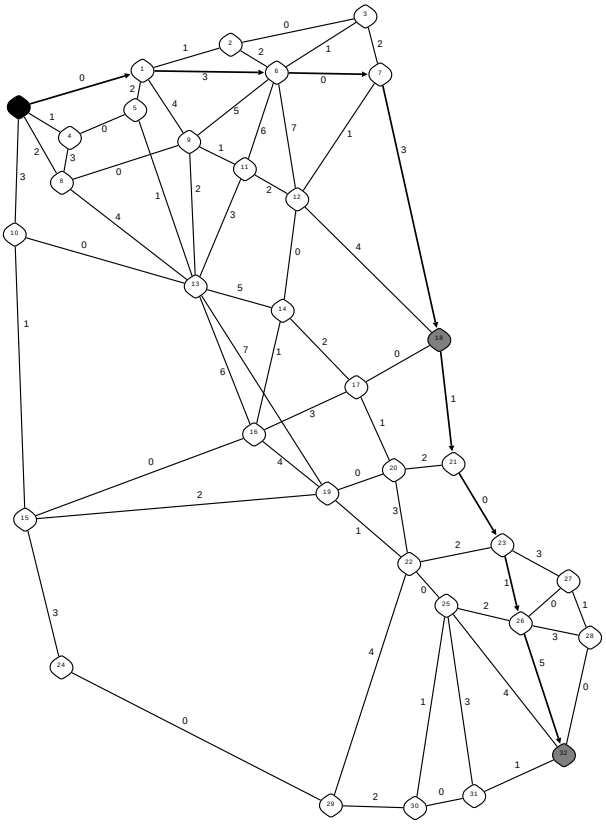


**Fig. 3.** ITALIAN-NET - An example link numbering and a multicast path.

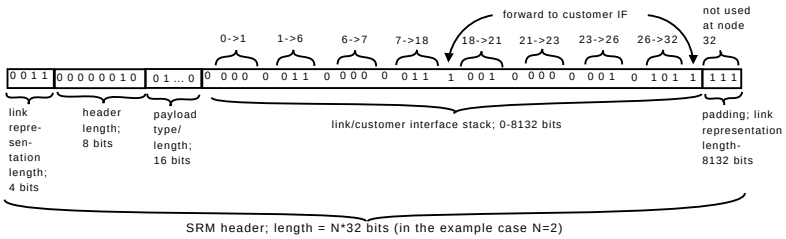
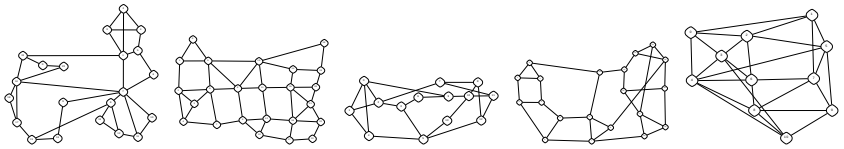



**Fig. 4.** SRM header corresponging to the path in Fig. 3 at the first core router.

smallest amount of bits in packet headers) possible, so that any *path* (sequence of link numbers) starting from a given node is still unique. Our approach has no problems if we connect two already numbered networks, and furthermore, *does not need routing or forwarding tables.*



(a) UNINETT (b) ATT (c) NSFNET (d) ARPANET (e) COST 239

**Fig. 5.** Test topologies.

We assign only one number for each link (instead of each interface). This approach enables the destination core router of a path to send a response to the source via the same path the packet traveled to it. For example, active measurement applications should benefit from this approach.

The distributed link numbering problem that our algorithm implicitly solves differs from the distributed graph edge coloring problem (see *e.g.* [11]) in one key aspect: we typically need to color (number) only one link at a time, while edge coloring algorithms are optimized for coloring an existing network that has no colored links in the beginning. Hence we can use a much simpler algorithm than the most sophisticated coloring algorithms.

The most common situation is the following: a core router notices a new link. In this case the adjacencent routers first synchronize their Link State Databases (LSDBs). Then both routers determine from their LSDBs whether each router having smaller ID has already numbered its adjacent links. If yes, then the router (one of the adjacent routers) determines from its LSDB the smallest number not yet assigned to any of its own core links, or the adjacent router's (over the newly detected link) core links [3] , and assigns this number for the new link. This means that if several links are detected simultaneously, our algorithm proceeds step-by-step from the core router having the smallest ID to the core router with largest ID. After each step the router that numbered its adjacent links floods the resulting numbers (as well as the adjacent node IDs) to other routers inside a routing protocol (*e.g.* OSPF or IS-IS). Then the router having the next smallest ID can number its adjacent links. If after the numbering there are number conflicts (*i.e.* one or more routers have the same number on two or more links) they are resolved in the order given by node IDs. An example of the result of automatic link numbering is given in Fig. 3.

A classical theorem by Vizing says that the edges of a graph of maximum degree $\Delta$ can always be colored with at most $\Delta+1$ colors. However, our algorithm sometimes needs more colors (numbers). Hence we simulated the performance of our algorithm in the real core networks of Fig. 5, in ITALIAN-NET, in a 10000-node grid graph, as well as in a 1000-node Waxman-graph generated with parameters $\alpha = 0.15, \beta = 0.2$ (see [14]), yielding diameter 3. We assigned the numbers in random order for the links. Table 1 gives the results. From the

[3] This default number selection policy aims to minimize overhead. To avoid topology leaking, the number selection should be configured to be "sparser".

**Table 1.** Mean amount of numbers used by our algorithm compared to minimum.

| | | | |
|---|---|---|---|
| ITALIAN-NET: 1 | UNINETT: 1 | ATT: 1.17 | NSFNET: 1 |
| ARPANET: 1 | COST 239: 1.14 | Grid (10000 nodes): 1.6 | Waxman (1000 nodes): 1.10 |

table we can conclude that our algorithm rarely requires additional bits for link representation compared to optimal $(\Delta + 1)$ numbering. Hence we believe that the simplicity of our numbering algorithm outweighs the small overhead increase it produces (compared to optimal numbering).

## 3 Routing and Forwarding in SRM

### 3.1 Overview

The main idea of our multicast paradigm can be illustrated by means of analogy. Consider the *Open Vehicle Routing Problem* [2], in which one has a depot that hosts one or more delivery cars. The depot is located at a certain network node, while the network links represent roads. Also, one needs to send some physical objects from the depot to customers (a subset of network nodes) using the cars. The problem is called *Open* if the cars do not need to return to the depot after delivering the objects to customers. Now if we do not limit the number of cars or their capacities, and we allow loops in the car routes, we have the essence of our multicast paradigm. Let us now give two more formal definitions, as well as two observations that intuitively motivate our approach.

**Definition 1**. *Minimum cost source-rooted multicast paths* (SRMP) *problem*: Find the set of directed paths (possibly containing loops) starting from source node, so that each destination (core router next to a destination) is a part of at least one of the paths. Find the set in such a way that the sum cost of links that are used by the paths is minimal.

**Definition 2**. *Delay-constrained minimum cost source-rooted multicast paths* (DC-SRMP) *problem*: SRMP with the restriction that all paths must have a given delay or less.

**Observation 1**. With symmetric link costs optimal solution to SRMP has a cost that is at most twice the cost of the optimal Steiner arborescence covering the same set of destination nodes (cost increase factor 2). With asymmetric link costs the cost increase factor is at most $D$ (nrof core routers next to destinations).
**Proof**. *Symmetric link costs*: Take the cost-optimal Steiner tree (cost-optimal arborescence with backward links) and traverse it with Depth First Search (DFS), adding each traversed link (also backward links) to a path. Now any link of the optimal Steiner tree will be traversed at most twice. *Asymmetric link costs*: Select the lowest cost path to each destination. □

**Observation 2**. A *feasible* solution to DC-SRMP exists whenever a feasible solution to the corresponding delay-constrained Steiner tree (arborescence) problem exists. Also, in this case the cost increase factor is at most $D$
**Proof**. Any branch of a Steiner arborescence can be enforced by means of a source routed path. Hence, given the cost-optimal Steiner arborescence that satisfies delay constraints, SRM can send packets to any destination via the same path that is used in the arborescence. □

The main point of these proofs is that cost-optimal solutions to SRMP and DC-SRMP *never* have a higher cost than $D*$(cost of the corresponding cost-optimal Steiner arborescence[4]), and in case of symmetric link costs and no delay constraints, cost-optimal solutions to SRMP *never* have a higher cost than $2*$(cost of the corresponding cost-optimal Steiner arborescence). The latter result shows that theoretically SRM produces lower cost than IP multicast. This will be verified empirically in Section 4. It is well known that both Steiner arborescence problem as well as SRMP are $\mathcal{NP}$-hard. However, many practical (optimal as well as heuristic) solution methods are known for both.[5]

### 3.2 SRM header and forwarding of SRM packets

A key concept of our paradigm is the header structure of a multicast packet (SRM header). An example of this header is given in Fig. 4. An SRM header contains the following fields in the given order:

1. **Link representation length** in bits; 4 bits (maximum core router degree $2^{2^4-1} = 32768$)
2. **Header length** in 32-bit words; 8 bits (maximum path length $\lfloor 8132/16 \rfloor - 1 = 507$ hops)
3. **Payload type/length**; 16 bits (similar to EtherType field in Ethernet header)
4. **Link/customer interface stack + padding**; variable length, altered by routers on the path

The link representation length tells each router of the path how many bits are used in this header for expressing link numbers. By default the number of bits is determined based on the number of bits required to represent the largest link number of the used path. However, in order to hide the maximum node degree and to gain faster memory operations in core routers, the router vendor/network operator might decide to use a larger number of bits than required (most likely 7 bits) to represent links. In this case the automatic link numbering protocol could select the numbers more sparsely, allowing more randomness to the padding.

The link/customer interface stack is a sequence of bits that indicates for every router of the path (1) if the router should forward the packet to customer interface(s), and (2) the next link (if any) where the packet should be forwarded.

[4] Provided by LIPSIN
[5] See *e.g.* web site neo.lcc.uma.es/radi-aeb/WebVRP/

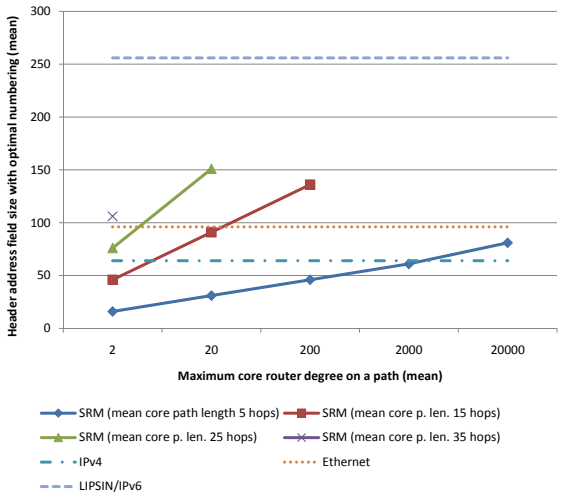



**Fig. 6.** Address sizes in headers. Data points shown only if the false positive rate of LIPSIN (see [1]) on the path was on average below 20%. Mean core router degree was assumed to be half of the maximum degree.

There is one important aspect about the padding. It must begin with a bit sequence that *does not* correspond to a link number at the last core router of the path. This way the last router can determine the end of path. To avoid topology leaking, the padding should be selected in an intelligent way (future work) with sparse link numbering.

Whenever an SRM packet arrives to a core router, the default action is to look at the first bit in the link/customer interface stack. If it is 1, then the router forwards the payload to customer interface(s). Let the link representation length be 7 bits. Now the router looks at the next 7 bits in the stack. If they correspond to a link number, the router moves these 7 bits prepended with a 0 to the end of the header, and forwards the packet to the link.

### 3.3 Scalability of SRM

Now let us evaluate the scalability of SRM. In most real networks the maximum node degree is less than 30 (see *e.g.* [1] and Rocketfuel). Hence we typically need at most five bits (maximum degree $2^5 - 1 = 31$) for link representation. Let the average Autonomous System (AS) level path length be 3 [13], and the average core path length inside a single AS 15. In this case the link/customer interface stack length would be $5 * 45 + 46 = 271$ bits. Hence (since the Internet diameter is decreasing [12]) SRM could be used without a notable overhead increase (compared to *e.g.* IPv6 or LIPSIN) even if the whole Internet were a single domain. Fig. 6 presents mean link/customer interface stack size as a function of mean path length and maximum node degree.

## 4 Delay and cost of SRM, IP multicast, and LIPSIN - empirical evaluation

In this section we will compare the delay and cost provided by SRM, IP multicast, and LIPSIN. To understand the following discussion, the reader should have some knowledge of linear optimization [17], and more specifically, how it is usually applied to Steiner problems [7, 8]. Since explicit delay constraints are arguably rare, we will focus explicitly on cost and implicitly on delay. In other words, we optimize for cost and present the resulting costs and maximum delays for each of the paradigms.

We used Integer Linear Programming (for an introductory treatment, see *e.g.* [17]) and Gurobi-solver[6] to find an optimal solution to the minimum-cost Steiner arborescence problem, as well as an *upper bound* for SRMP by using a Steiner arborescence formulation that disallows branching outside source. This formulation was chosen because it can be easily implemented, and can be solved relatively quickly by using the well-known cut strategies from Steiner arborescence formulations (see [7, 8]).

To use this method in a real network, it would be possible to install the ILP solver (Gurobi in our case) to each rendezvous point, formulate the ILP program in another program (for example, written in C) whenever a JOIN or a PART message arrives to the rendezvous point, call Gurobi from the C-program, and install/modify the returned `multicast group` → `source routed multicast path` mappings in the kernel of the rendezvous point. In case of a JOIN, it might be useful to let the rendezvous point first install the shortest path to the destination (to facilitate fast response time), and solve the ILP off-line. Once the ILP program returns, the rendezvous point could re-configure its `multicast group` → `source routed multicast path` mappings.

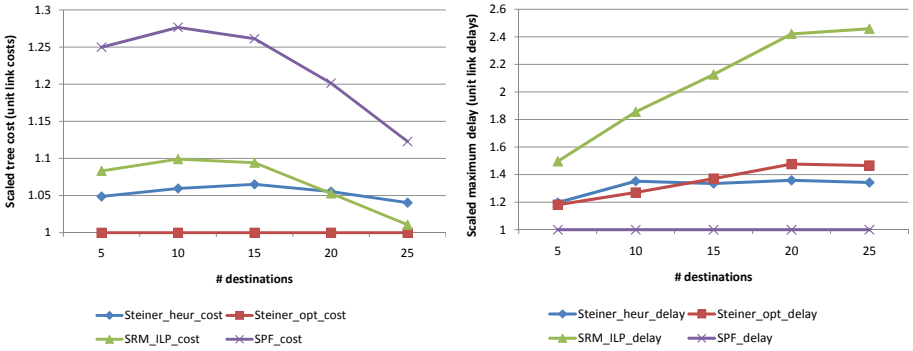
Our test networks (besides ITALIAN-NET) are presented in Fig. 5. They are commonly used in the literature. We also tested grid graph of 16 and 64 nodes. The cost increase factors resulting from our upper bound ILP formulation of SRMP compared to the optimal Steiner arborescence solutions with asymmetric link costs are presented in Table 2 (averages over 100 test runs). The number of destinations was 0.25∗(number of nodes) rounded to the nearest integer. More detailed test results for ITALIAN-NET are presented in Fig. 7.

From subfigures 7(a) and 7(b) it can be seen that with unit link costs and delays the SRMP solutions (SRM_ILP) have on average a lower cost than the

[6] www.gurobi.com

**Table 2.** Cost increase factor of SRM (upper bound ILP formulation) compared to optimal Steiner arborescences. Asymmetric costs selected randomly from interval (1,100).

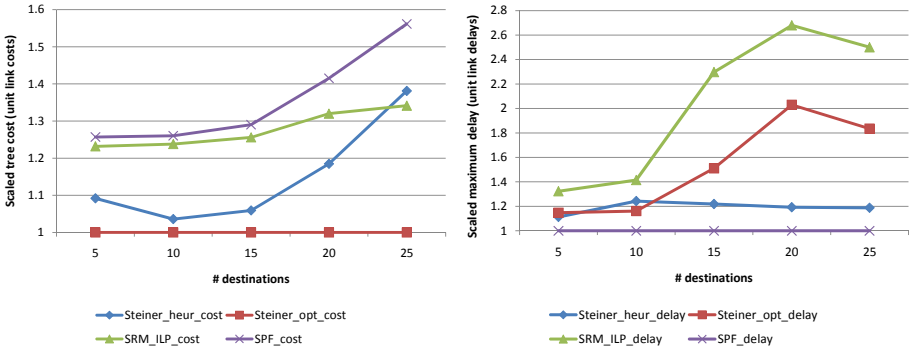
| | | | |
|---|---|---|---|
| ITALIAN-NET: 1.21 | UNINETT: 1.47 | ATT: 1.18 | NSFNET: 1.24 |
| ARPANET: 1.09 | COST 239: 1.09 | Grid (16 nodes): 1.15 | Grid (64 nodes): 1.30 |




(a) Cost; unit delays and costs




(b) Delay; unit delays and costs




(c) Cost; Asymmetric delays and costs selected randomly from interval (1,100)




(d) Delay; Asymmetric delays and costs selected randomly from interval (1,100)

**Fig. 7.** Test results for ITALIAN-NET. No explicit delay constraints are used.

optimal ones provided by IP multicast (SPF), and are at worst 10% higher than the optimal ones facilitated by LIPSIN (Steiner_opt).

From subfigures 7(c) and 7(d) it can be seen that the cost gap with *asymmetric* random link costs between SRMP and Steiner tree solutions is about 22%-35%. With many subscribers the SRMP solutions have a lower cost than the Steiner tree solutions obtained with the approximation algorithm of [3] (Steiner_heur). However, the delays of SRM tend to be quite long. In other words, it seems that solutions to SRMP would rarely be feasible solutions to DC-SRMP. However, it should be noted that in all tests the set of destinations was chosen randomly, which is the worst case for SRM.

## 5 Conclusions

We presented an automatic link numbering algorithm, and a related multicast paradigm called Source Routed Multicast (SRM). We compared SRM to IP

multicast, as well as to a new proposal, LIPSIN. Overall, our analysis revealed that SRM is very competitive against these approaches.

## 6 Acknowledgements

The authors would like to thank Marko Luoma and Aleksi Penttinen for their comments.

## References

1. P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, P. Nikander: *LIPSIN: line speed publish/subscribe inter-networking*, In: Proc. of SIGCOMM, 2009.
2. A. N. Letchford, J. Lysgaard, R. W. Eglese: *A Branch-and-Cut Algorithm for the Capacitated Open Vehicle Routing Problem*, In: J. of the Operational Research Society, Vol. 58, Nr. 12, pp. 1642-1651(10), 2007.
3. L. Kou, G. Markowsky, L. Berman: *A fast algorithm for Steiner trees*, In: Acta Informatica, Vol. 15, Nr. 2, pp. 141-145, 1981.
4. W.-T. Chen, P.-R. Sheu, Y.-R. Chang: *Efficient multicast source routing scheme*, In: Computer Communications, Vol. 16, Iss. 10, pp. 662-666, 1993.
5. T.-S.P. Yum, M. S. Chen: *Multicast source routing in packet-switched networks*, In: IEEE Transactions on Communications, Vol. 42, Nr. 234, pp. 1212-1215, 1994.
6. X. Zhang, J. Y. Wei, and C. Qiao: *Constrained Multicast Routing in WDM Networks with Sparse Light Splitting*, In: J. of Lightwave Technology Vol. 18, Nr. 12, pp. 1917-1927, 2000.
7. T. Koch and A. Martin: *Solving Steiner tree problems in graphs to optimality*, In: Networks, Vol. 32, Iss. 3, pp. 207-232, 1998.
8. I. Ljubic, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, and M. Fischetti: *Solving the prize-collecting Steiner tree problem to optimality*, In: Proc. of ALENEX workshop, SIAM, 2005.
9. V. Sharma, F. Hellstrand: *Framework for Multi-Protocol Label Switching (MPLS)-based Recovery*, RFC 3469, 2003.
10. M. Doar, I. Leslie: *How Bad is Naive Multicast Routing?*, In: Proc. of INFOCOM, 1993.
11. L. Barenboim, M. Elkin: *Distributed* $(\Delta + 1)$*-coloring in linear (in* $\Delta$*) time*, In: Proc. of the 41st Symposium on Theory of Computing, 2009.
12. J. Leskovec, J. Kleinberg, C. Faloutsos: *Graph evolution: Densification and shrinking diameters*, In: ACM Transactions on Knowledge Discovery from Data (TKDD), Vol. 1, Iss. 1, 2007.
13. J. Zhao, P. Zhu, X. Lu, L. Xuan: *Does the Average Path Length Grow in the Internet?*, In: Proc. of ICOIN, 2007.
14. E.W Zegura, K.L Calvert, S. Bhattacharjee: *How to model an internetwork*, In: Proc. of INFOCOM, 1996.
15. B. Yang, P. Mohapatra: *Edge router multicasting with MPLS traffic engineering*, In: Proc. of ICON, 2002.
16. A. Boudani, B. Cousin: *A New Approach to Construct Multicast Trees in MPLS Networks*, In: Proc. of ISCC, 2002.
17. D. Bertsimas, J.N. Tsitsiklis: *Introduction to Linear Optimization*, Athena Scientific 1997.