Multicast Algorithms in Service Overlay Networks

Dario Pompili*, Luca Lopez†, Caterina Scoglio‡

*Broadband and Wireless Networking Laboratory
Georgia Institute of Technology, Atlanta, GA 30332, USA

†Dipartimento di Informatica e Sistemistica
University "La Sapienza," Rome 00184, Italy

‡Department of Electrical and Computer Engineering
Kansas State University, Manhattan, KS 66506, USA

e-mail: *dario@ece.gatech.edu (corresponding author), †lopez.luca@gmail.com, ‡caterina@eece.ksu.edu

***Abstract*— Overlay routing has been proposed to enhance the reliability and performance of IP networks, since it can bypass congestion and transient outages by forwarding traffic through one or more intermediate overlay nodes. In this paper, two algorithms for multicast applications in service overlay networks are presented. The first is tailored for source specific applications and builds virtual source rooted multicast trees to allow one node in the multicast group to send data to the other member nodes. The second is tailored for group shared applications and constructs a virtual shared tree among group members. Their objective is to achieve traffic balancing on the overlay network so as to avoid traffic congestion and fluctuation, which cause low network performance. To address these problems, the algorithms actively probe the underlay network and compute virtual multicast trees by dynamically selecting the least loaded available paths on the overlay network. The low-complexity algorithms are shown through simulation experiments to lead to time and resource saving.**

***Index Terms*— Overlay Networks, Multicast, Optimization.**

I. Introduction

OVERLAY routing has been proposed to enhance the reliability and performance of IP networks, since it can bypass congestion and transient outages by forwarding traffic through one or more intermediate overlay nodes [1][2][3]. While much of the past research in overlay networking has focused on techniques for building overlay networks and evaluating their performance (e.g., [4][5]), in this paper we present two algorithms to build virtual multicast trees on an overlay network for as many applications as live video, software and file distribution, replicated database, web site replication, videoconference, distributed games, etc.

We consider two layers of network infrastructure: the *native network*, which includes end-systems, routers, links, and the associated routing functionality, and provides best-effort datagram delivery between its nodes; the *virtual overlay network*, which is formed by a subset of the native layer nodes interconnected through overlay links to provide enhanced services. Overlay links are *virtual* in the sense that they are IP tunnels over the native network.

The first proposed algorithm is called *DIstributed Multicast algorithm for Internet Resource Optimization* (DIMRO). It builds virtual source rooted multicast trees for source specific applications. DIMRO takes into account the virtual link available bandwidth in order to avoid traffic congestion and fluctuation, which cause low network performance. The idea is to keep the average link utilization of the overlay network low, by fairly distributing data flows among the least loaded links. The second algorithm is called *DIstributed Multicast algorithm for Internet Resources Optimization in Group Shared applications* (DIMRO-GS). It constructs a virtual shared tree for group shared applications by connecting each member node to all the other member nodes with a source rooted tree computed using DIMRO. Both DIMRO and DIMRO-GS algorithms offer service differentiation, i.e., provide QoS at application-layer without IP-layer support.

II. DIMRO: DIstributed Multicast algorithm for Internet Resource Optimization

By simply minimizing the cost of multicast trees without taking into account the overlay link bandwidth availability, traffic congestion and fluctuation might occur in the network, causing low network performance. In order to tackle this problem, DIMRO follows a different approach. The virtual multicast source rooted tree is built by choosing the paths in the overlay network that are least loaded. This way, DIMRO achieves load balancing and an optimal distribution of resources, which leads to maximize the number of multicast trees that can be set up. Let us consider a multirate multicast scenario where receivers ask for different rates. If K channels with rate $\{w_1, .., w_K\}$ are used in the layering scheme, then K cumulative rate $L_1, .., L_K$ are available. Let s be the source node, and let us assume that M receivers belong to the multicast group. DIMRO proceeds as it follows.

Step 0: Receivers are ordered from the highest rate to the lowest one, so as to have an ordered set of M receivers $\{r_1, .., r_M\}$, with requested cumulative rates $F_1 \geq .. \geq F_M$ ($F_j \in \{L_1, .., L_K\}$, $j = 1, .., M$). Receivers from r_1 to r_M are progressively connected to the source node. This way, receiver r_i can reuse resources already exploited on paths from s to $r_1, .., r_{i-1}$. Let $S_{path}(s, r_k)$ be the set of all feasible paths from source s to receiver r_k. A path $p(s, r_k)$ from s to r_k is feasible if $b_{uv} \geq F_k$ for all its links, where b_{uv} is the available bandwidth on virtual link (u, v), which represents the achievable bit rate on the associated underlay path.

Step k, k=1,..,M: DIMRO chooses path $\overline{p(s,r_k)} \in S_{path}(s,r_k)$ that minimizes the function $f\{p(s,r_k)\} = \sum_{\{(u,v)\in p(s,r_k)\}} \frac{a_{uv}}{(1-\rho_{uv})^{\gamma}}$, where $p(s,r_k) \in S_{path}(s,r_k)$, and:

1) ρ_{uv} is the utilization of link (u,v), and it is defined as

$$\rho_{uv} = \frac{B_{uv} - (b_{uv} - F_k)}{B_{uv}} = \frac{B_{uv} - \beta_{uv}}{B_{uv}}, \quad (1)$$

where B_{uv} represents the total bandwidth capacity of virtual link (u,v), F_k is the bandwidth exploited on link (u,v) by receiver r_k, and $\beta_{uv} = b_{uv} - F_k$ is the residual bandwidth on link (u,v). These measurements are acquired using *active probing* on the path in the underlay network associated with link (u,v).

2) The exponent $\gamma = \gamma(|\mathcal{V}|, |\mathcal{E}|, \overline{F}, \overline{B})$ is a function of the number of nodes $|\mathcal{V}|$ and links $|\mathcal{E}|$ composing the overlay network, the average rate $\overline{F}$ requested by receivers, and the average bandwidth $\overline{B}$ of the overlay links.

3) The binary variable a_{uv} is equal to 0 if link (u,v) already belongs to the tree, and 1 otherwise.

Computational complexity: The complexity of DIMRO is $O(M \cdot |\mathcal{V}| \cdot |\mathcal{E}|)$ since it builds the virtual multicast tree by computing for as many as M times the spanning tree using the Bellman-Ford algorithm, whose complexity is $O(|\mathcal{V}| \cdot |\mathcal{E}|)$.

III. DIMRO-GS: DIMRO IN GROUP SHARED APPLICATIONS

DIMRO-GS (Group Shared DIMRO) is the extension of DIMRO to the multicast shared tree case. If there are M group members, the virtual shared tree can be set up by building M virtual source rooted trees, each one having a different group member as root and all the other members belonging to the tree, not necessarily as leaves. Each source rooted tree is built using DIMRO. Let us assume that each group member has the same bandwidth requirement, i.e., $W_k = \widetilde{W}$, $k = 1,..,M$. In this case, the first step of DIMRO is skipped. When M source rooted trees are computed, the virtual shared tree is completed.

Computational complexity: The complexity of DIMRO-GS is $O(M^2 \cdot |\mathcal{V}| \cdot |\mathcal{E}|)$ since it runs DIMRO M times.

IV. PERFORMANCE EVALUATION

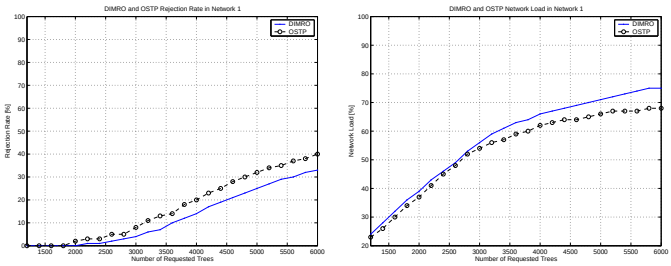
A random overlay network has been generated following the Waxman's model [6], according to which network nodes are randomly distributed across a Cartesian coordinate grid. Virtual links are added to the graph modeling the overlay network by considering all possible pairs (u,v) of nodes and by using the following probability function,

$$P_e(u,v) = \beta \cdot \exp\left(-\frac{d_{uv}}{\alpha \cdot L}\right), \quad (2)$$

where $P_e(u,v)$ is the existence probability of a virtual link between nodes u and v, d_{uv} is the Euclidean distance between u and v, L is the maximum possible distance between a pair of nodes, and α and β are parameters belonging to the range (0,1]. A high α value increases the number of connections to nodes further away, while a high β value increases the node degree. Unlike the original Waxman's model, we assume that each virtual link is *bi-directional*. The bandwidth capacity B_{uv} of each virtual link (u,v) is randomly generated using a uniform distribution with mean $\overline{B}$. Two different one hundred node random overlay networks are generated, according to Tab. I. Network 2 has a higher number of links than Network 1, according to the probability function (2). The average bandwidth in both networks is $\overline{B} = 100\ Mbps$.

TABLE I
NETWORK PARAMETERS

	α	β	$\overline{B}$	$nodes$
Overlay Net. 1	0.2	0.4	$100\ Mbps$	100
Overlay Net. 2	0.3	0.6	$100\ Mbps$	100



(a) (b)

Fig. 1. DIMRO and OSTP *Rejection Rate* and *Network Load* in Net. 1

DIMRO is compared to the optimal solution of the Steiner tree problem [7] when all virtual link costs are equal to 1. If every link has a cost equal to 1, the *minimum-weight tree* that spans all group members is the multicast tree that uses the least number of links. The minimum-weight tree is found by solving the flow formulation of the Steiner tree problem. We implemented the Integer Linear Programming (ILP) problem in AMPL [8] and solved it with CPLEX. Two metrics are used to compare the competing algorithms:

1) The *Rejection Rate*, defined as

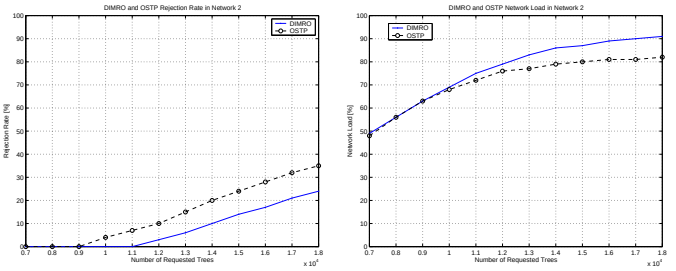
$$\textit{Rejection Rate} = \frac{\textit{Number of Rejected Trees}}{\textit{Number of Requested Trees}}, \quad (3)$$

2) the *Network Load* $\overline{\rho}$, defined as

$$\overline{\rho} = \frac{\sum_{\{(u,v)\in\mathcal{E}\}} \rho_{uv}}{|\mathcal{E}|}. \quad (4)$$

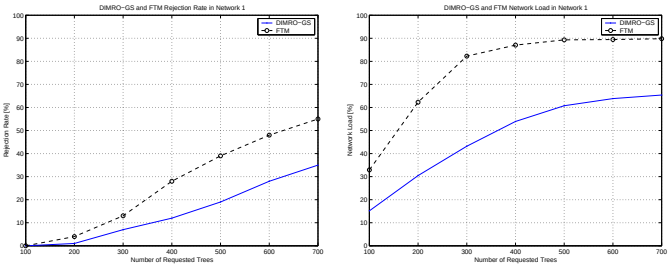
For each simulation campaign several experiments have been run to ensure a 95% relative confidence intervals smaller than 5%. Starting from a completely unloaded Waxman overlay network, it is requested to build a fixed number of source rooted trees (*Number of Requested Trees*). Multicast groups are sequentially randomly generated, and their members (source and receivers) are randomly chosen among the network nodes. The number of receivers for each multicast group is uniformly distributed from 5 to 15, and the bandwidth request of each receiver is uniformly distributed from 0.1 to 2 $Mbps$.

Figure 1(a) shows that the DIMRO *Rejection Rate* in Network 1 is lower than the *Rejection Rate* of the Optimal solution of the Steiner Tree Problem (OSTP) with $c_{uv} = 1$. The *Rejection Rate* is approximately the same until the *Number*



(a) (b)

Fig. 2. DIMRO and OSTP *Rejection Rate* and *Network Load* in Net. 2



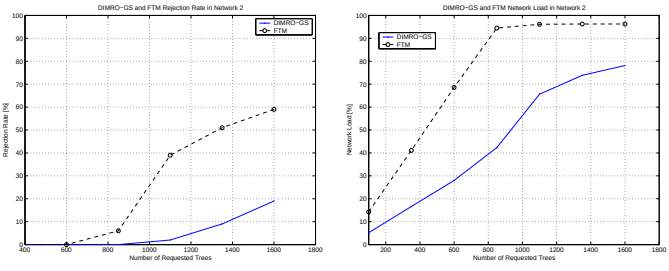
(a) (b)

Fig. 3. DIMRO-GS and FTM *Rejection Rate* and *Network Load* in Net. 1

of Requested Trees is less than 2500. When the *Number of Requested Trees* overcomes this threshold, DIMRO has a lower *Rejection Rate*. Both DIMRO and OSTP *Rejection Rates* increase when the *Number of Requested Trees* increases, since the same bottlenecks occur. These bottlenecks depend on the underlay network topology and cannot be avoided, although the DIMRO *Rejection Rate* is lower because bottlenecks occur later. Figure 1(b) shows that the DIMRO *Network Load* is the same as the OSTP *Network Load* until the *Rejection Rate* is the same. Then, since DIMRO rejects less trees, its *Network Load* becomes higher than the OSTP one. In Network 2, the DIMRO *Rejection Rate* is significantly lower than the OSTP *Rejection Rate* (Fig. 2(a)). This is because less *unavoidable* bottlenecks[1] exist. Since Network 2 has a higher number of links, the number of possible paths between two nodes increases; thus, it is easier for DIMRO to avoid bottlenecks. A lower *Network Load* (Fig. 1(b) and Fig. 2(b)) with a higher *Rejection Rate* (Fig. 1(a) and Fig. 2(a)) proves that OSTP does not efficiently use the available network resources.

To evaluate the performance of the DIMRO-GS and FTM algorithm [9], we slightly adapted the two metrics previously introduced to the group shared case. Simulation results for both Network 1 and Network 2 (Fig. 3(a) and Fig. 4(a)) show that when the *Number of Requested Trees* increases, the FTM *Rejection Rate* becomes significantly higher than the one of DIMRO-GS, since FTM uses a higher amount of network resources. In fact, when the *Number of Requested Trees* increases, the FTM *Network Load* becomes significantly higher than the DIMRO-GS *Network Load* (Fig. 3(b) and Fig. 4(b)). This implies that network resources *saturate* later when

[1]An *unavoidable* bottleneck is a bottleneck that depends on the underlay network topology and not on the algorithm run on the overlay network.



(a) (b)

Fig. 4. DIMRO-GS and FTM *Rejection Rate* and *Network Load* in Net. 2

DIMRO-GS is used, which causes a lower *Rejection Rate*. Figures 4(a) and 4(b) show that the DIMRO-GS algorithm achieves a lower *Rejection Rate* and *Network Load* in Network 2. Note that in Network 1 (Fig. 3(a) and Fig. 3(b)) bottlenecks do not allow to efficiently exploit all network resources.

V. Conclusions and Future Work

Two algorithms for multicast applications in service overlay networks were presented. The first builds virtual source rooted multicast trees for source specific applications; the second constructs a virtual shared tree for group shared applications. Their objective is to achieve traffic balancing on the overlay network so as to avoid traffic congestion and fluctuation, which cause low network performance. The algorithms actively probe the underlay network and compute virtual multicast trees by dynamically selecting the least loaded available paths on the overlay network. Future research will focus on dynamic multicast groups, and on the interactions between overlay and underlay networks.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under grant no. ANI-0219829.

References

[1] H. Zhang, J. Kurose, and D. Towsley, “Can an overlay compensate for a careless underlay?” in *Proceedings of IEEE INFOCOM 2006*, Barcelona, Spain, Apr. 2006.

[2] Y. Zhu, C. Dovrolis, and M. Ammar, “Dynamic overlay routing based on available bandwidth estimation: A simulation study,” *To appear in the Computer Networks Journal*, 2006.

[3] S. Y. Shi and J. S. Turner, “Routing in overlay multicast networks,” in *Proceedings of IEEE INFOCOM 2002*, New York, NY, USA, June 2002.

[4] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, “Resilient overlay networks,” in *Proceedings of ACM Symposium on Operating Systems Principles*, Banff, Canada, Oct. 2001.

[5] A. Nakao, L. Peterson, and A. Bavier, “A routing underlay for overlay networks,” in *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.

[6] B. M. Waxman, “Routing of multipoint connections,” *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, December 1988.

[7] R. T. Wong, “A dual ascent approach for Steiner tree problems on a directed graph,” *Mathematical Programming*, vol. 28, pp. 271–287, 1984.

[8] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Cole Publishing Co., 2002.

[9] C. P. Low and N. Wang, “On Finding Feasible solutions to Group Multicast Routing Problem,” *IEICE Transactions on Communications*, vol. E85-B, no. 1, pp. 268–277, January 2002.