Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 489–505
computer communications
www.elsevier.com/locate/comcom

Multicast algorithms in service overlay networks ☆

Dario Pompili [a,*], Caterina Scoglio [b], Luca Lopez [c]

[a] Rutgers, The State University of New Jersey, Department of Electrical and Computer Engineering, Piscataway, NJ 08854, USA
[b] Kansas State University, Department of Electrical and Computer Engineering, Manhattan, KS 66506, USA
[c] University of Rome La Sapienza, Department of System Engineering, 00184 Rome, Italy

Available online 29 August 2007

Abstract

Overlay routing enhances the reliability and performance of IP networks since it can bypass network 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 algorithm is tailored for source-specific applications such as live video, software and file distribution, replicated database, web site replication, and periodic data delivery; it builds a virtual source-rooted multicast tree to allow one member in the multicast group to send data to the other members. The second algorithm is tailored for group-shared applications such as videoconference, distributed games, file sharing, collaborative groupware, and replicated database; it constructs a virtual shared tree among group members. The objective of both algorithms is to achieve traffic balancing on the overlay network so as to avoid traffic congestion and fluctuation on the underlay network, which cause low 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. This way, network resources are optimally distributed and the number of multicast trees that can be setup is maximized. Both algorithms can offer service differentiation, i.e., provide QoS at application-layer without IP-layer support. The low computational complexity of the proposed algorithms leads to time and resource saving, as shown through extensive simulation experiments.

Keywords: Overlay networks; Multicast; Mathematical programming/optimization; Multirate layering

1. Introduction

Overlay routing has been proposed to enhance the reliability and performance of IP networks since it can bypass network congestion and transient outages by forwarding traffic through one or more intermediate overlay nodes [34,41,42]. Service overlay network is an effective means to address end-to-end Quality of Service (QoS), plaguing the current Internet, and to facilitate the creation and deployment of value-added Internet services such as VoIP, Video-on-Demand, and other emerging QoS-sensitive services. While much of the past research in overlay networking has focused on techniques for building and reconfiguring overlay networks, and evaluating their performance (e.g., [3,17,30,37]), 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, periodic data delivery, videoconference, distributed games, file sharing, and collaborative groupware. Moreover, we propose a low complexity multirate layering algorithm to support the virtual multicast algorithms by accommodating heterogeneous receiver requested bit rates.

Throughout this paper, 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; and a *virtual overlay network*, which is formed by a subset of the native layer nodes interconnected

☆ This work was supported by the National Science Foundation (NSF) under Grant No. ANI-0219829.
* Corresponding author. Tel.: +1 732 445 6400x202.
E-mail address: pompili@ece.rutgers.edu (D. Pompili).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.08.023

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 *DIfferentiated service Multicast algorithm for Internet Resource Optimization (DIMRO)*. It builds virtual source-rooted multicast trees for source-specific applications. DIMRO takes the virtual link available bandwidth into account to avoid traffic congestion and fluctuation on the underlay network, which cause low performance. The objective 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 *DIfferentiated service Multicast algorithm for Internet Resource 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. To support these algorithms, a novel low complexity layering algorithm to accommodate heterogeneous receiver requested bit rates is also proposed, which makes DIMRO and DIMRO-GS native multirate multicast algorithms on overlay networks.

Both DIMRO and DIMRO-GS algorithms offer *service differentiation*, i.e., provide QoS at application-layer without IP-layer support. Consequently, multicast group members with less stringent QoS requirements manage to reuse resources already exploited by members with more stringent requirements. This results in a better utilization of network bandwidth, and in an improved QoS as perceived by multicast group members.

To summarize, the main contributions of this work are:

(1) Development of QoS-aware multicast algorithms on overlay networks, which are able to provide application-layer QoS for multicasting without IP-layer support.
(2) Design of low-complexity overlay multirate algorithms to leverage the network bandwidth resources and improve QoS as perceived by multicast group members.

The remainder of the paper is organized as follows: in Section 2, we provide a brief background on multicasting, while in Section 3 we review the relevant literature on overlay networks. In Section 4, we formulate the logical channel rate assignment problem and describe DIMRO, while in Section 5 we introduce DIMRO-GS. In Section 6, we show numerical results through extensive simulation experiments. Finally, in Section 7, we draw the main conclusions and point future work.

2. Background on multicasting

In unicast transmissions, the sender transmits data to a single receiver and, if multiple receivers want to receive the same data content, the sender has to transmit multiple copies of data. In multicast transmission, conversely, the sender transmits only one copy of data that is delivered to multiple receivers. This allows to efficiently exploit Internet resource in as many applications as live video, software and file distribution, replicated database, web site replication, periodic data delivery, videoconference, distributed games, file sharing, and collaborative groupware. One of the most challenging objective in multicasting is to minimize the amount of network resources utilized to compute and setup multicast trees [32,33]. In multicast communication, the routing problem is to find the *minimum-weight tree* that spans all the nodes in the multicast group [15,36,40]. Multicast communication can be classified into two types, i.e., *source specific* and *group shared*. In source-specific multicast communication, only one node in the multicast group sends data, while all the other member nodes receive data. In group-shared multicast communication, each node in the multicast group wants to send/receive data to/from member nodes. A tree that spans all member nodes is called *multicast tree*. Consequently, based on the communication strategy, multicast trees can be classified into two types, i.e., *source-rooted trees* and *shared trees*. A source-rooted tree has the source node as root, and is optimized for source-specific multicast communications, whereas a shared tree is optimized for group-shared communications, and connects each group member with all the other group members.

The classical optimization problem in source-specific multicast communications is the *Steiner tree Problem in Networks* (SPN) [2], whose objective is to find the least-cost tree connecting the source and the group of destinations with the minimum total cost over all links. If each destination has a bandwidth requirement, then the problem is to find the *least-cost* tree that complies with the bandwidth requirements on each path from the source to the receiver. It can be shown that both these problems are at least as complex as the *geometric connected dominating set* problem, which is proven to be NP-complete [21,28]. Hence, both these problems are NP-complete, and efficient algorithms to solve these problems in polynomial time attain only approximate solutions in most cases [2,36].

In group-shared communication, one of the earliest protocol to build a shared tree was the Core-Based Protocol (CBT) [5,6] and its enhanced version (CBTv2) [4], which connect each member node to a *core node* using bi-directional shortest paths. The main drawback is the traffic concentration occurring in the core node. To avoid traffic concentration, a shared tree can be computed by finding as many source-rooted trees as the number of member nodes, each of them with all the other member nodes as leaves. The FTM algorithm [29,38], Feasible solutions using adapted Takahashi–Matsuyama (TM) algorithm, follows this approach. The source-rooted tree connects the root node with the member node by using the path with the greatest capacity. The bottleneck of a path is characterized as the link with the minimum available bandwidth, and the path bandwidth capacity is defined as the available bandwidth of its bottleneck. The path with the greatest

bandwidth capacity is called the *widest path*. The FTM algorithm uses information obtained by the Breadth First Search procedure (BFS) [20]. For each member node, the BFS procedure builds a tree rooted at the member node and connects each network node with this member node by exploiting the *widest path paradigm criterion*.

While traditional multicast algorithms were integrated in the IP layer, often with no QoS support, in this paper we present two QoS-aware algorithms to build virtual multicast trees on an overlay network. 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 [34,41,42]. In particular, we devise algorithms to construct source-routed and shared multicast trees by explicitly taking into account the available bandwidth of virtual links in the overlay network, whereas most of the past research on overlay networking has focused on techniques for building and reconfiguring overlay networks, and evaluating their performance (e.g., [3,17,30,37]).

3. Related work

There has been extensive work on building multicast algorithms on top of overlay networks, like in [12,13,25,31], or even commercial schemes like BitTorrent or Digital Fountain. However, neither of these algorithms have addressed the QoS requirements of multicast group. Conversely, our present contribution is focused on integrating QoS in overlay multicasting. Specifically, we also consider the common case in current network topologies where the bottleneck link is likely to be on the last hop near the clients, which are thus served at different rates.

Several two-tier overlay multicast network infrastructures have been recently proposed as feasible solutions to support scalable inter-domain multicast services for real-time applications. In [35,26], a set of dedicated devices called multicast service nodes (MSNs) are assumed to be distributed in the network, typically collocated at the sites of a selected number of access routers. It was shown in [35] that it is useful to balance bandwidth usage among MSNs for each multicast session. This way, the overlay multicast network is able to support a larger number of concurrent multicast sessions with good service qualities. Earlier proposals for building an overlay multicast tree typically aim to optimize the backbone tree that contains MSNs only [7,35]. They assume that the MSN placement solution is given and the access subtrees rooted at each particular MSN are also given. The routing protocols proposed in [7,35] addressed three constrained spanning tree problems: (i) minimize the maximum end-to-end delay within the set of MSNs subject to the interface bandwidth constraint at each MSN; (ii) minimize the average end-to-end delay within the set of MSNs subject to the interface bandwidth constraint at each MSN; and (iii) balance the interface bandwidth usage at each MSN subject to the bound on the maximum end-to-end delay. Although the approaches proposed in [7,35,26] are sound and very interesting, in these papers each MSN is assumed to function not only as a multicast server for the access router it is collocated with, but also as a transit point for replicating and forwarding real-time streaming traffic to other MSNs or access routers in the overlay multicast network by means of routine unicast mechanisms. This means that, according to these approaches, the topology of the overlay multicast network is considered as a fully connected mesh, which may not hold in general.

In [23], QUEST, a QoS assUred composEable Service infrasTructure is proposed, which can provide both QoS assurances under multiple QoS constraints, and load balancing in service overlay networks. Service composition is performed by the service composer using a service provisioning protocol, which is designed based on a network-centric client-service model. In [8], key functions of Skype, a VoIP client developed by KaZaa in 2003, are studied by analyzing its network traffic. Because of the overlay peer-to-peer paradigm used, Skype allows its users to place voice calls and send text messages to other users of Skype clients almost seamlessly across Network Address Translations (NATs) and firewalls. In [27], it is assumed that each autonomous system in the Internet has one or more Overlay Brokers (OBs) that cooperate with each other to form an overlay service network and provide overlay service support for overlay applications. Specifically, a QoS-aware routing protocol for overlay network to balance the overlay traffic among OBs and overlay links is presented. Interestingly, [27] introduces a general unified framework that may be a desirable alternative to application-specific overlays. However, the routing solution proposed is tailored for unicast transmissions, and does not address host multicasting. In [16], the bandwidth provisioning problem in service overlay networks is mathematically formulated, and the critical issues concerning cost recovery in deploying and operating the value-added services are analyzed. The framework accounts for several factors such as Service Level Agreement (SLA), service QoS, traffic demand distributions, and bandwidth costs. Moreover, analytical models and approximate solutions are developed for both static and dynamic bandwidth provisioning.

As briefly reviewed in this section, most research work on service overlay networks has addressed problems in wide-area service composition such as the fault-resilience problem, the adaptability problem, and the resource contention problem to find a multimedia service path. The QoS consistency and load partition issues for composing service path in ubiquitous computing environments were also addressed in several overlay network projects. However, still much needs to be done to support generic QoS provisioning for host multicasting in service overlay networks. Hence, in this paper we tackle this problem by proposing multicast algorithms on overlay networks with generic virtual topologies.

4. DIMRO: Differentiated service Multicast algorithm for Internet Resource Optimization

In this section, we present DIMRO, an efficient algorithm to build source-rooted trees for source-specific applications. In particular, in Section 4.1, we formulate the channel rate assignment problem for the exact determination of channel rates in a multirate multicast environment to accommodate heterogeneous receiver requested bit rates. In Section 4.2, we describe how DIMRO works in *non-QoS-aware* overlay networks and show how this algorithm manages to keep the probability of bottleneck occurrence low. Finally, in Section 4.3, we describe the main improvements in DIMRO when a *QoS-aware*[1] overlay network is considered, and the full set of features of the algorithm is described and analyzed.

4.1. Optimal channel rate assignment problem

In source-specific communications, multicast sessions may have a large number of receivers with heterogeneous reception capacities. To accommodate this heterogeneity, we propose a novel *layering scheme*. In a layering scheme, data transmission through the network takes place over *logical channels*, i.e., a sender/receiver can simultaneously transmit/receive data on multiple channels. In a multirate multicast scenario, data on logical channel n is transmitted at rate w_n. Receivers subscribe to the layers cumulatively, i.e., if a receiver subscribes to layer j, it also subscribes to layers $\{1, 2, \ldots, j-1\}$ and receives data at the cumulative rate $L_j = \sum_{n=1}^{j} w_n$. Two of the most well-know multirate schemes are proposed in [9,10].

In [9], channel rates are determined to minimize the total *completion time*, defined as the sum of all receivers' completion time, i.e., the time that a receiver needs to download the file. With M receivers and K channels, the algorithm proposed in [9] has a computational complexity $O(M^3 \cdot K)$. Conversely, in the layering scheme introduced in [10], channel rates are computed as follows:

$$w_j = \begin{cases} b & \text{if } j = 1 \\ \sum_{t=1}^{j-1} w_t & \text{if } j > 1, \end{cases} \quad (1)$$

where b is a base rate. Cumulative rates computed by the layering scheme in [10] match requested rates worse than those calculated by the layering scheme in [9]. However, the layering scheme in [10] allows dynamic multicast groups, i.e., a receiver can join the group and start downloading the file at any time. This is possible since the transmission on each channel is *cyclic*, which in turn is possible because channel rates are given as in (1). However, the objective function of the layering scheme in [10] does not meet properties of *fairness*, as defined in [24], i.e., not all receivers are offered the same service level. For this reason, differently from [10], in this paper we determine channel rates by minimizing the average ratio between requested rate and cumulative rate subscribed by a receiver. This way, our objective function meets properties of fairness, as it will be clear in the following. We formulate hereafter the channel rate assignment problem, while we refer the reader to the Appendix for the detailed description of an efficient algorithm to solve this problem in polynomial time. This layering scheme is exploited by our overlay multicast algorithms.

In a layering scheme a file is divided into segments. Copies of the same segment are transmitted through different logical channels. Fig. 1 shows an example of a 'nested' layering scheme for multicasting. A data stream of length B is transmitted over three channels to three receivers with maximum rates $W_1 = 1$, $W_2 = 3$, and $W_3 = 4$. The channel rates are set to $w_1 = 1$, $w_2 = 1$, and $w_3 = 2$. Receiver 1 subscribes to channel 1, receiver 2 to channels 1 and 2, and receiver 3 to all the three channels. In this example, the sender partitions the data stream into 4 equal-sized segments 'a', 'b', 'c', and 'd'. First segment 'a' is transmitted over channel 1, 'b' over channel 2, and 'c' and 'd' over channel 3. These concurrent transmissions finish at the same time $B/4$. At this time, only receiver 3 has received the entire stream. The sender then transmits segment 'c' over channel 1 and 'd' over channel 2 concurrently. When these transmissions complete (at time $B/2$), receiver 2 has received all four segments, but receiver 1 still has not received segments 'b' and 'd'. These are thus then transmitted over channel 1. At time B, receiver 1 completes.

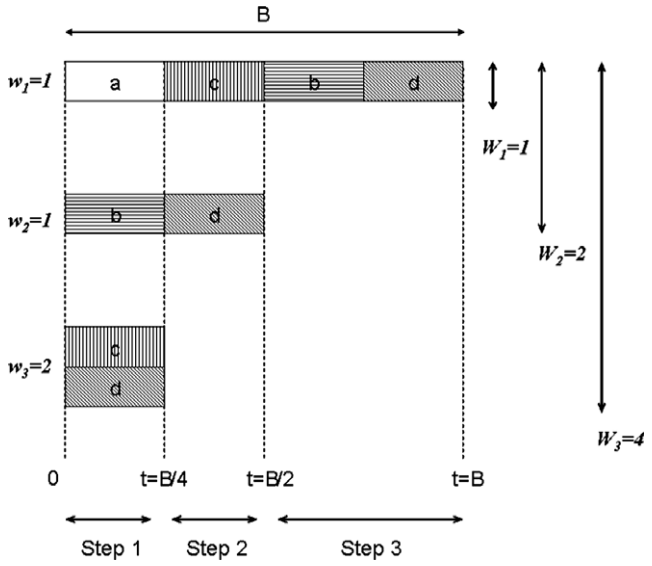


Fig. 1. Example of a 'nested' layering scheme for multicasting.

[1] In this paper, we define as *QoS-aware* overlay network a subset of the native layer nodes interconnected through overlay links that are able to provide *differentiated services*, i.e., different end-to-end QoS guarantees to the applications. Such differentiation may be achieved by relying on different topologies of the underlay networks, i.e., an overlay link may be associated with a long underlay path for delay-insensitive low priority class traffic, and with a short path for delay-sensitive high priority class traffic. This service differentiation at the overlay network brings high flexibility in QoS provisioning even when the underlay network provides only best effort service, i.e., does not explicitly implement mechanisms to support QoS.

In this example, by using three channels for three receivers, each receiver completes in its *ideal completion time*, i.e., each receiver receives data at the demanded rate.

The number of logical channels used in a layering scheme, indicated as K hereafter, is a critical parameter that should be chosen by trading off different contrasting issues. By increasing the number of channels K, in fact, the number of file segments grows. The higher the number of file segments, the higher the complexity for a receiver to rebuild the file. Furthermore, the more channels, the more copies of the same segment are transmitted through different channels, i.e., the higher the *overhead* associated with the layering scheme (in the above example, only segment 'a' is transmitted once, while segments 'b' and 'c' are transmitted twice, and segment 'd' three times). On the other hand, the higher K, the easier to match the requested rates by assigning logical channels. For applications with a large number of receivers and high variability in rates, however, it is unrealistic to provide enough channels to match requested rates of all receivers. Hence, the optimal number of channels K^* should be determined to achieve as much benefit for the receivers as possible, without incurring in excessive overhead.

Let us consider a set of M receivers with N different requested bit rates $W_1 < \ldots < W_N$, with $N \leqslant M$. Let us assume that the sender can setup a maximum number of logical channels, $K^{\max}$, which depends on its computation resources. Thus, the optimal number of channels K^* the sender will use is

$$K^* = \min(K^{\max}, N). \tag{2}$$

If the number of logical channels K^* is equal to the number N of different requested rates, i.e., $N \leqslant K^{\max}$, then the logical channel rates $(w_1, .., w_{K^*})$ can be trivially chosen as

$$w_i = W_i - W_{i-1}, \quad i = 1, \ldots, N, \quad W_0 = 0. \tag{3}$$

In this case, if a receiver with requested rate W_j subscribes the set of channels $\{1, 2, \ldots, j\}$, then it will receive data at the cumulative rate $L_j = \sum_{n=1}^{j} w_n = W_j$, which exactly represents the requested rate at which it would receive data.

Conversely, if the number of channels K^* is lower than the number of requested rates N, i.e., $K^{\max} < N$, as it is the case in a realistic scenario, then the cumulative rates $L_1, \ldots, L_{K^*}$ can be determined by solving the optimization problem **P**, whose objective is to achieve fairness among all receivers, i.e., minimize the average ratio of requested rate and cumulative rate subscribed by a receiver. This way, each receiver is granted a bit rate whose *distance* to its requested ideal rate is proportional to the ideal rate itself.

If we assume that each receiver is assigned a weight γ_m, $m = 1, \ldots, M$, then $\eta_1, \ldots, \eta_N$ in the objective function of **P** are N weights such that η_n, $n = 1, \ldots, N$, is the sum of the weights of those receivers asking for rate W_n. If, for the sake of simplicity, all receivers are considered to have the same weight $\gamma = 1$, then η_n represents the number of receivers asking for rate W_n. We can now cast the optimal logical channel rate assignment problem.

P: **Optimal Logical Channel Rate Assignment Problem**

$$\text{Given :} \quad K^*; N; W_n, \quad n = 1, \ldots, N$$

$$\text{Find :} \quad L_k, y_{n,k}, \quad k = 1, \ldots, K^*, \quad n = 1, \ldots, N$$

$$\text{Maximize :} \quad \phi(L_1, \ldots, L_{K^*}) = \sum_{n=1}^{N} \frac{\eta_n}{W_n} \cdot F_n$$

Subject to : (4)

$$F_n = \sum_{k=1}^{K^*} L_k \cdot y_{n,k} \leqslant W_n, \quad n = 1, \ldots, N; \tag{5}$$

$$L_k \in \{W_1, \ldots, W_N\}, \quad k = 1, \ldots, K^*; \tag{6}$$

$$y_{n,k} \in \{0, 1\}, \quad k = 1, \ldots, K^*, \quad n = 1, \ldots, N; \tag{7}$$

$$\sum_{k=1}^{K^*} y_{n,k} = 1, \quad n = 1, \ldots, N. \tag{8}$$

As in (5), F_n defines the cumulative rate obtained by a receiver with requested rate W_n. Constraints (7) and (8) imply that $F_n \in \{L_1, \ldots, L_{K^*}\}, n = 1, \ldots, N$. Once the cumulative rates F_n are determined by solving problem **P**, then the corresponding channels rates are easily computed as

$$w_j = L_j - L_{j-1}, \quad j = 1, .., K^*, \quad L_0 = 0. \tag{9}$$

In the Appendix, we present an efficient algorithm to exactly solve problem **P** in polynomial time, which is characterized by a complexity $O(K^* \cdot N^2)$. We also provide an example for the sake of clarity.

4.2. DIMRO in non-QoS-aware overlay networks

By simply minimizing the cost of multicast trees without taking the overlay link bandwidth availability into account, traffic congestion and fluctuation might occur in the underlay network, causing low performance. In fact, since many multicast trees may use the same set of virtual links, these links could be overloaded while other links in the network could remain under-utilized. DIMRO follows a different approach to tackle this problem. 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 lead 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 $\{w_1, \ldots, w_{K^*}\}$ are used in the layering scheme, then K^* cumulative rate $L_1, \ldots, L_{K^*}$ are available, as derived in **P**. 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. At the end of this step we have an ordered set of M receivers $\{r_1, \ldots, r_M\}$, with requested cumulative rates $F_1 \geqslant F_2 \geqslant \ldots \geqslant F_M (F_j \in \{L_1, \ldots, L_{K^*}\}, j = 1, \ldots, M)$. Receivers from r_1 to r_M are connected to the source node progressively. This way, receiver r_j can reuse resources

already exploited on paths from the source to receivers $r_1,\ldots,r_{j-1}$. These receivers, in fact, ask for rates $F_1 \geqslant F_2 \geqslant \ldots \geqslant F_{j-1} \geqslant F_j$.

Let $S_{\text{path}}(s,r_j)$ be the set of all feasible paths from s to r_j. A path $p(s,r_j)$ from source s to receiver r_j is feasible if $b_{uv} \geqslant F_j$ for all its links, where b_{uv} is the available bandwidth on overlay link (u,v).

Step j, $j=1,\ldots,M$: The algorithm chooses the path $p(s,r_j) \in S_{\text{path}}(s,r_j)$ that minimizes the following function

$$f\{p(s,r_j)\} = \sum_{\{(u,v)\in p(s,r_j)\}} \frac{a_{uv}}{(1-\rho_{uv})^{\alpha}}, \quad p(s,r_j) \in S_{\text{path}}(s,r_j). \tag{10}$$

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

$$\rho_{uv} = \frac{B_{uv} - (b_{uv} - F_j)}{B_{uv}} = \frac{B_{uv} - \beta_{uv}}{B_{uv}}, \tag{11}$$

where B_{uv} represents the total bandwidth capacity of virtual link (u,v), F_j is the bandwidth exploited on link (u,v) by receiver r_j, and $\beta_{uv} = b_{uv} - F_j$ is the residual bandwidth of link (u,v). Notice that ρ_{uv} is calculated after that b_{uv} has been decremented by F_j. These measurements are acquired using techniques such as *active or reactive probing* on the underlay network. Examples of these probing protocols are the MIT RON (Resilient Overlay Networks) [3,18], whose design goal is to allow end-hosts and applications to cooperatively gain improved reliability and performance from the Internet by examining the condition of the Internet between themselves and the other nodes. On the other hand, in [30], the authors argue that designing overlay services to independently probe the Internet is an untenable strategy. Instead, they propose PLUTO (PlanetLab Underlay Topology Services for Overlay Networks), a shared routing underlay that overlay services query, and posit that this underlay must adhere to two high-level principles. First, it must take cost (in terms of network probes) into account. Second, it must be layered so that specialized routing services can be built from a set of basic primitives. These principles lead to an underlay design where lower layers expose large-scale, coarse-grained static information already collected by the network, and upper layers perform more frequent probes over a narrow set of nodes.

(2) The exponent $\alpha = \alpha(|\mathcal{V}|,|\mathcal{E}|,\overline{F},\overline{B})$ in (10) is a function of the number of nodes $|\mathcal{V}|$ and links $|\mathcal{E}|$, the average rate $\overline{F}$ requested by receivers, and the average network link bandwidth $\overline{B}$. According to accurate empirical observations (described hereafter) supported by extensive simulation results, the model for the exponent α has been chosen as

$$\alpha(|\mathcal{V}|,|\mathcal{E}|,\overline{F},\overline{B}) = a\cdot\exp\left\{-b\cdot\frac{|\mathcal{E}|}{|\mathcal{V}|\cdot(|\mathcal{V}|-1)}\right\}\cdot\exp\left\{-c\cdot\frac{\overline{F}}{\overline{B}}\right\}. \tag{12}$$

Coefficients a, b, and c have been determined by solving the following minimization problem of the quadratic error between the exponent model in (12) and the experimental results,

$$\text{Minimize}: \sum_{i=1}^{I}[\alpha_i - \alpha(|\mathcal{V}|_i,|\mathcal{E}|_i,\overline{F}_i,\overline{B}_i)]^2 \quad a\in[0,3], b\in[0,+\infty). \tag{13}$$

Starting from a set of chosen points $\Gamma = \{(|\mathcal{V}|_i,|\mathcal{E}|_i,\overline{F}_i,\overline{B}_i), i=1,\ldots,I\}$, the optimum exponent α_i was experimentally obtained for each point $(|\mathcal{V}|_i,|\mathcal{E}|_i,\overline{F}_i,\overline{B}_i)\in\Gamma$ of the set. The coefficient a has been chosen belonging to the interval [0,3] after an accurate tuning of the model. Values of coefficients a, b, c obtained by solving the minimization problem (13) are $a=3$, $b=3.9$, and $c=16.9$.

Let us now go back to the α model in (12) to explain why this formula, with its variable aggregations $\overline{F}/\overline{B}$ and $|\mathcal{E}|/(|\mathcal{V}|(|\mathcal{V}|-1))$, is suitable to model the exponent in (10). By decreasing the value of the exponent α, the length of paths found by DIMRO is reduced because the difference between the metric of loaded links and the metric of unloaded links is reduced. It could be necessary to reduce the length of paths used by the multicast tree when the average rate $\overline{F}$ requested by the receivers grows, or when the average network bandwidth $\overline{B}$ decreases. In fact, the higher the value of $\overline{F}$, or the lower the value of $\overline{B}$, the more resources on each overlay link are consumed by a single path. Thus, short paths have to be preferred. Simulations show that when the ratio $\overline{F}/\overline{B}$ increases, the value of the optimal exponent α decreases exponentially.

As far as $|\mathcal{V}|$ and $|\mathcal{E}|$ are concerned, if the number of links $|\mathcal{E}|$ decreases, there are less paths in the network, and it becomes more important maximizing the resource utilization. By increasing the value of the exponent α, the difference between the metric of loaded links and unloaded links increases. If the number of nodes $|\mathcal{V}|$ grows but the number of links $|\mathcal{E}|$ remains the same, the exponent increases because the number of paths decreases. Simulations show that when the ratio $|\mathcal{E}|/(|\mathcal{V}|(|\mathcal{V}|-1))$ decreases, the value of the optimal exponent α grows exponentially.

(3) The binary variable a_{uv} is equal to 0 if virtual link (u,v) already belongs to the tree, otherwise it is 1. If the set $S_{\text{path}}(s,\ r_j)$ is not empty, let $p(s,r_j)$ be the path from source s to receiver r_j that minimizes the function in (10). On each link (u,v) belonging to $p(s,r_j)$, if $a_{uv}=0$, the value of the available bandwidth is not updated, and those bandwidth resources already exploited by a path $p(s,r_t)$, with $F_t \geqslant F_j$, are used. Conversely, if $a_{uv}\neq 0$, then new resources must be used, and the new value of the available bandwidth becomes $b'_{uv} = b_{uv} - F_j$. Then, the binary variable a_{uv} is set to 0 in order not to consider the cost of link (u,v) for those paths that will exploit it in the future.

Let us point out that at *Step* j, $j=1,\ ..,\ M$, the optimal path $p(s,r_j)\in S_{\text{path}}(s,r_j)$ can be found using a shortest-path algorithm (as Dijkstra or Bellman–Ford algorithm), where the length of each link in the network is set to

$$d_{uv} = \begin{cases} \frac{a_{uv}}{(1-\rho_{uv})^{\alpha}} & \text{if } b_{uv} \geqslant F_j \\ \infty & \text{if } b_{uv} < F_j \quad \text{or} \quad (u,v) \notin \mathcal{E}. \end{cases} \tag{14}$$

In this work we use the Bellman–Ford algorithm [14], which finds a spanning tree of the shortest paths from the source to all other nodes of the graph. Therefore, the path $p(s,n)$ from node s to node n, solution of (10), is the one that minimizes the function

$$\sum_{\{(u,v)\in p(s,n)\}} d_{uv}. \tag{15}$$

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 $O(|\mathcal{V}| \cdot |\mathcal{E}|)$.

4.3. DIMRO in QoS-aware overlay networks

In this section, the DIMRO algorithm is extended to leverage the differentiated service provided by QoS-aware overlay networks, i.e., able to provide *differentiated services* (different end-to-end QoS guarantees to the applications). Let us consider a multirate multicast scenario where receivers ask for the same data content but different rates and different QoS. In this scenario, the extended DIMRO algorithm allows a receiver to reuse the bandwidth already exploited by receivers asking for higher service classes without any extra cost for the network. By building a path from source s to a receiver r, the DIMRO algorithm can reuse some sub-paths already exploited by higher service class receivers. Thus, receiver r obtains a *better* QoS, and the network saves resources because bandwidth already exploited by other receivers is reused for receiver r. DIMRO in a QoS-aware overlay network proceeds into steps, described hereafter.

Step 0: Receivers are ordered according to their service class and rate. Let us consider a set $\mathcal{R}$ made up of M receivers. Let $\mathcal{CL} = \{\mathrm{cl}_1, \ldots, \mathrm{cl}_L\}$ be the set of service classes requested by receivers, where cl_1 is the highest service class and cl_L is the lowest one. The set of receivers $\mathcal{R}$ is partitioned into $L = |\mathcal{CL}|$ subsets,

$$\mathcal{R} = \bigcup_{i=1}^{L} \mathcal{R}_i. \tag{16}$$

Subset $\mathcal{R}_i$ is made up of those receivers asking for service class cl_i, $i = 1, \ldots, L$. Receivers in each subset $\mathcal{R}_i$, are ordered from the highest rate to the lowest one. Let us consider a partitioned and ordered set $\mathcal{R}$ made up of M receivers. Let r_j^i be the receiver j in the subset $\mathcal{R}_i$ with requested rate $F_j^i \cdot |\mathcal{R}_i|$ stands for the cardinality of $\mathcal{R}_i$, and i represents the associated service class.

Step i, $i = 1, \ldots, L$: DIMRO connects source s with all those receivers asking for service class cl_i. Initially, the binary variables a_{uv} are set to 1 for each overlay link (u,v). For receiver r_j^i, $j = 1, \ldots, |\mathcal{R}_i|$, a_{uv} are set to 0 for all those overlay links (u,v) already used by receivers that ask for higher service class and higher or equal rate then r_j^i. In fact, resources already exploited by the tree on these links could be used by r_j^i without any added cost.

Let $b_{uv}(\mathrm{cl}_i)$ be the available bandwidth of link (u,v) for the service class cl_i. A path $p(s, r_j^i)$ from source s to a receiver r_j^i asking for cumulative rate F_j^i is feasible if $b_{uv}(\mathrm{cl}_i) \geqslant F_j^i$ for all its links. DIMRO chooses that feasible path $\overline{p(s, r_j^i)}$ from s to r_j^i that minimizes the following function,

$$f\{p(s, r_j^i)\} = \sum_{\{(u,v)\in p(s,r_j^i)\}} \frac{a_{uv}}{(1-\rho_{uv}(\mathrm{cl}_i))^{\alpha}}, \quad p(s, r_j^i) \in S_{\mathrm{path}}(s, r_j^i), \tag{17}$$

where $S_{\mathrm{path}}(s, r_j^i)$ is the set of feasible paths from s to r_j^i.

Exponent α is defined as in (12), and utilization $\rho_{uv}(\mathrm{cl}_i)$ of the overlay link (u,v) is calculated as

$$\rho_{uv} = \frac{B_{uv}(\mathrm{cl}_i) - [b_{uv}(\mathrm{cl}_i) - F_j^i]}{B_{uv}(\mathrm{cl}_i)}, \tag{18}$$

where $B_{uv}(\mathrm{cl}_i)$ is the bandwidth capacity of overlay link (u,v) for the class cl_i, and F_j^i is the cumulative rate requested by r_j^i.

For each overlay link (u,v) belonging to path $\overline{p(s, r_j^i)}$, if $a_{uv} = 0$ then the available bandwidth $b_{uv}(cl_i)$ is not updated and the new path uses those resources already exploited by other receivers. Conversely, if $a_{uv} \neq 0$, then new resources have to be exploited and the new value of the available bandwidth becomes $b'_{uv}(\mathrm{cl}_i) = b_{uv}(\mathrm{cl}_i) - F_j^i$. Then, the binary variable a_{uv} is set to zero.

The DIMRO algorithm determines the path $\overline{p(s, r_j^i)}$ by using a modified shortest-path algorithm. Common shortest-path algorithms could choose one of the possible paths with zero cost from s to n, but this shortest path, as it will be clear in the following example, might not be correct. To determine a correct path the algorithm proceeds as it follows.

Starting from node s, the first link (u,v) belonging to $\overline{p(s, r_j^i)}$ with $a_{uv} \neq 0$ is found. Among all those paths in the tree exploiting a bandwidth equal or greater than F_j^i and passing through node u, it is chosen the one that uses the highest service class. Let us indicate this path with $\tilde{p}_u$. The new path $\overline{\tilde{p}(s, r_j^i)}$ from s to r_j^i is the concatenation of the sub-path from s to node u, belonging to path $\tilde{p}_u$, and the sub-path from node u to r_j^i, belonging to $\overline{p(s, r_j^i)}$.

Example. Fig. 2 shows a multicast tree on the overlay network where receiver r_1 (node 6) asks for a rate $F_1 = 2$ and a QoS mapped into service class A, receiver r_2 (node 8) asks for a rate $F_2 = 3$ and a QoS mapped into service class B, and receiver r_3 (node 7) asks for a rate $F_3 = 2$ and a QoS mapped into service class C. In this example, we assume A to be the highest service class, while C to be the lowest. Path from source s, node 1, to receiver r_1 is the first to be built. Receiver r_2 asks for a service class B but a rate $F_2 = 3 > F_1 = 2$, thus the binary variables a_{uv} are not set to

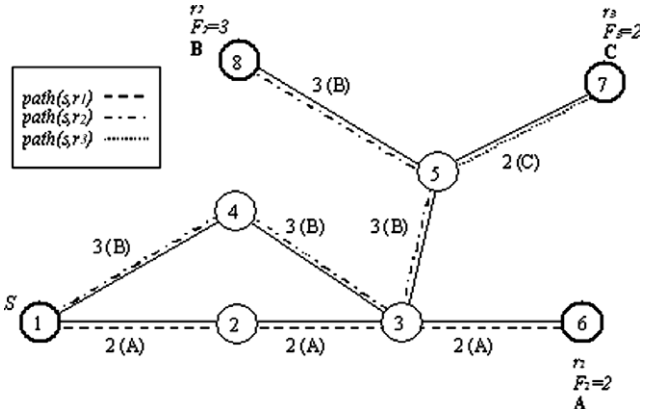


Fig. 2. Correct path computation.

0 on links of path from node 1 to node 6. Path from node 1 to node 8 is built. Let $p(s,r_i)$ be the path from source s to receiver r_i. Receiver r_3 asks for a service class C and a rate $F_3 = F_1 = 2 < F_2 = 3$ thus, on each link of paths $p(s, r_1)$ and $p(s,r_2)$, a_{uv} are set to 0 because r_3 can reuse resources already exploited by other receivers. By using a common shortest-path algorithm, path $\bar{p} = \{1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 7\}$ might be found from source s to receiver r_3 (node 7) because paths $\{1 \rightarrow 2 \rightarrow 3 \rightarrow 5\}$ and $\{1 \rightarrow 4 \rightarrow 3 \rightarrow 5\}$ have both cost equal to zero. The point is that path $\bar{p}$ cannot be taken into consideration since link (3, 5) belongs only to the path $\{1 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 8)\}$ ($p(s,r_2)$ in Fig. 2) and not to the path $\{1 \rightarrow 2 \rightarrow 3 \rightarrow 6\}$ ($p(s,r_1)$ in Fig. 2). For the mentioned reason, it should be clear that the correct path is $\{1 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 7\}$. Once the shortest-path algorithm has determined a path from the source s, node 1, to receiver r_3, node 7, the last node of the zero cost path is determined, node 5 in Fig. 2. All paths connecting the source with those receivers that ask for a higher or equal service class and a higher or equal requested rate than r_3 are visited (in Fig. 2 path from s to r_1 and from s to r_2). Only path from s to $r_2$has a sub-path $\{1 \rightarrow 4 \rightarrow 3 \rightarrow 5\}$ from s to node 5. Thus, path $\{1 \rightarrow 4 \rightarrow 3 \rightarrow 5\}$ is chosen among all path with zero cost from s to node 5. Finally, the correct paths $\{1 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 7\}$ from s to r_3 is selected.

5. DIMRO-GS: Differentiated service Multicast algorithm for Internet Resource Optimization in Group-shared Applications

In this section, we present DIMRO-GS, an efficient algorithm to build shared trees for group-shared multicast communications. A shared tree is a generalization of a source-rooted tree. In a source-rooted tree the source node multicasts data to a set of receiver nodes, whereas in a shared tree each member node can multicast data to all the other member nodes (videoconference is probably the most common multicast application requiring a shared tree, but many others applications are conceivable such as distributed games, file sharing, collaborative groupware, replicated database, etc.)

In Section 5.1, we explain how DIMRO-GS works in non-QoS-aware overlay networks and show how this algorithm manages to keep the probability of bottleneck occurrence low. In Section 5.2, we point out the main improvements in DIMRO-GS when QoS-aware overlay networks are considered. In this last case, the full set of features characterizing DIMRO-GS is described and analyzed.

As briefly introduced in Section 1, a possible approach to build a shared tree is using the Core-Based Protocol (CBT) [5,6] or its enhanced version (CBTv2) [4]. The main drawback with this approach is the traffic concentration occurring in the core node. Thus, some links might be overloaded, specially in multicast applications with high bandwidth requirements. Moreover, multicast trees set up with this approach are quite easy to be computed, but are far from being optimized. Let us consider an extreme example to catch the inefficiency of this protocol: if two member nodes had a link connecting each other, still their traffic would pass through the core node, no matter the state of their common link.

To avoid traffic concentration a shared tree can be computed by separately finding M source-rooted trees (where M is the number of member nodes). The source-rooted tree i, $i = 1, \ldots, M$, has the member node n_i as root and all the other member nodes n_j, $j = 1, \ldots, M$ (with $j \neq i$), belonging to the tree, not necessarily as leaves. The FTM (Feasible solutions using adapted TM algorithm) algorithm, proposed in [29,38], follows this approach. The source-rooted tree i, $i = 1, \ldots, M$ connects the root node n_i with the member node n_j, $j = 1, \ldots, M$, $j \neq i$, by using the path with the greatest capacity from node n_i to node n_j. The bottleneck of a path is characterized as the used link with the minimum available bandwidth, and the path bandwidth capacity is defined as the available bandwidth of its bottleneck. The path with the greatest bandwidth capacity from node n_i to node n_j is called the *widest path*. The FTM algorithm uses information obtained by the Breadth First Search procedure (BFS) [20]. For each member node the BFS procedure builds a tree rooted at the member node and connecting each network node with this member node by exploiting the *widest path paradigm criterion*. After M BFS trees are built rooted at each group member n_i, the detailed pieces of information about the *widest path* connecting each pair of member nodes are available.

In the next step, FTM builds M multicast source-rooted trees, one for each group member n_i. As far as the member node n_i is concerned, paths with the maximum bandwidth capacity will be used by the source node n_i to join all the other group member. If two or more paths with the same bandwidth capacity are found at the same time, the one with the least cost is selected. In order to generate each multicast tree, two tables must be available, i.e., the *bandwidth capacity* table and the *path cost* table. These tables are both $M \cdot |\mathcal{V}|$ in size, where M is the number of members and $|\mathcal{V}|$ is the number of overlay nodes.

Computational complexity: The FTM computational complexity is dominated by the BFS complexity, which is

$O((|\mathcal{V}| + |\mathcal{E}|) \cdot |\mathcal{V}|^2)$. Thus, building M multicast trees by exploiting the FTM algorithm is $O(M \cdot (|\mathcal{V}| + |\mathcal{E}|) \cdot |\mathcal{V}|^2)$.

5.1. DIMRO-GS in non-QoS-aware overlay networks

In Section 4 the DIMRO algorithm has been presented. This algorithm can be modified to build multicast shared trees. We call this modified algorithm DIMRO-GS (Group Shared DIMRO). If there are M group members, the shared tree can be set up by building M source-rooted trees, each one having a different group member as root and all the other group members belonging to the tree, not necessarily as leaves. Each source-rooted tree can be built according to the DIMRO algorithm. Let us assume that each group member has the same bandwidth requirement, i.e., $W_j = \widetilde{W}$, $j = 1, \ldots, M$. In this case, the first step of the DIMRO algorithm, aimed at ordering receivers according to their bandwidth requirement, is skipped. When M source-rooted trees are computed, the shared tree is completed. From then on, each member group can send data to all the other members and receive data from all the other members.

Computational complexity: In the shared-tree case, if M is the number of group members, then M source-rooted trees have to be built. The computational complexity for each source-rooted tree by using the DIMRO algorithm is $O(M \cdot |\mathcal{V}| \cdot |\mathcal{E}|)$, as seen in Section 4.2; hence, the computational complexity of the DIMRO-GS algorithm becomes $O(M^2 \cdot |\mathcal{V}| \cdot |\mathcal{E}|)$.

It has been shown in the previous section that the computational complexity of the FTM algorithm is $O(M \cdot (|\mathcal{V}| + |\mathcal{E}|) \cdot |\mathcal{V}|^2)$. Since in most practical case $M \ll |\mathcal{V}|$, the computational complexity of the proposed algorithm is lower than the FTM complexity. Furthermore, FTM needs two tables of size $M \cdot |\mathcal{V}|$ that must be stored in memory, while in DIMRO-GS no table is required.

5.2. DIMRO-GS in QoS-aware overlay networks

In this section, the DIMRO-GS algorithm is extended for QoS-aware overlay networks. Let us consider a multicast scenario where members in a group-shared communication can ask for different QoS requirements. The extended DIMRO-GS algorithm allows a group member acting as receiver to reuse the bandwidth already exploited by group members asking for higher service classes, without any extra cost for the network. By building a path from a group member s, acting as source, to a group member r, acting as receiver, DIMRO-GS can reuse some sub-paths already exploited by group members, acting as receivers, with more stringent QoS requirements. Thus, group member r obtains a *better* QoS and the network saves resources because bandwidth already exploited by other group members is reused for group member r. The extended DIMRO-GS algorithm in a QoS-aware overlay network proceeds as follows.

Let us consider a set $\mathcal{R}$ made up of M group members. At the first step, group members are ordered from the highest service class to the lowest one. The set $\mathcal{R}$ is ordered per service class, i.e., if $i < j$, then member group n_i asks for a higher or equal service class then n_j. In the following, we provide a simple example to show how DIMRO-GS works.

Example. Let us consider a simple differentiated service overlay network with only four different classes of service: A, B, C, D. Let us assume that A is the highest service class, whereas D is the lowest one. Let us indicate a group member r with the couple $(F;C) = (\textit{Requested Rate};\textit{Service Class})$ representing the rate and service class requested by group member r. Let us consider the following multicast group: $\mathcal{R} = \{(1;B), (1;A), (1;B), (1;D), (1;C), (1;C)\}$. The algorithm proceeds as it follows.

Step 0: The multicast group is sorted per service class in a *lexicographic* order, as shown hereafter:

$$\mathcal{R}^* = \{(1;A), (1;B), (1;B), (1;C), (1;C), (1;D)\}.$$

Step i, $i = 1, \ldots, M$: DIMRO-GS uses the DIMRO algorithm to builds a source-rooted tree with the member node n_i as root and all the other member nodes belonging to the tree. At the beginning of *Step i*, the binary variables a_{uv} are initialized to 1 for each overlay link (u, v). Let cl_j be the service class requested by the group member n_j, $j = 1, \ldots, M$, $n_j \neq n_i$, acting as receiver, and let $b_{uv}(\mathrm{cl}_j)$ be the available bandwidth for the service class cl_j on link (u, v). A path $p(n_i, n_j)$ from node n_i to node n_j is feasible if $b_{uv}(\mathrm{cl}_j) \geqslant \widetilde{W}$ for all its links ($\widetilde{W}$ is the rate requested by each group member). For each group member n_j, $j = 1, \ldots, M$, $n_j \neq n_i$ DIMRO chooses that feasible path $p(n_i, n_j)$ from n_i to n_j that minimizes the following function

$$f\{p(n_i, n_j)\} = \sum_{\{(u,v) \in p(n_i, n_j)\}} \frac{a_{uv}}{(1 - \rho_{uv}(\mathrm{cl}_j))^{\alpha}}, \quad p(n_i, n_j) \in S_{\mathrm{path}}(n_i, n_j), \tag{19}$$

where $S_{\mathrm{path}}(n_i, n_j)$ is the set of all feasible paths from n_i to n_j. For each overlay link (u, v) belonging to path $p(n_i, n_j)$, if $a_{uv} = 0$, then the available bandwidth $b_{uv}(cl_j)$ is not updated, and the new path uses those resources already exploited by another member group, acting as receiver, that asks for a higher or equal service class. Conversely, if $a_{uv} \neq 0$, then new resources must be exploited and, thus, the new value of the available bandwidth becomes $b'_{uv}(\mathrm{cl}_j) = b_{uv}(\mathrm{cl}_j) - \widetilde{W}$. Then, variable a_{uv} is set to 0. In this way, the path from n_i to n_t, $t = j + 1$, .., M, can reuse resources already used on path $p(n_i, n_j)$ without any adding cost for the network.

6. Simulation results

6.1. Random network model

To ensure a fair evaluation of different routing algorithms, a random overlay network has been generated

according to the Waxman's model [39,11]. In the Waxman's model, network nodes are randomly distributed across a Cartesian coordinate grid. Virtual links are added to the graph by considering all possible pairs (u,v) of nodes and by using the probability function,

$$P_e(u,v) = \beta \cdot \exp\left(-\frac{d_{uv}}{\alpha \cdot D_{\max}}\right), \tag{20}$$

where $P_e(u,v)$ is the existence probability of a link between nodes u and v, d_{uv} is the Euclidean distance between the node u and the node v, $D_{\max}$ is the maximum possible distance between a pair of nodes, α and β are parameters in 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 [39], we assume that each link added to the network is bi-directional. The bandwidth capacity B_{uv} of each link (u,v) is randomly generated using an uniform distribution with mean bandwidth $\overline{B}$. Consequently, link (u,v) bandwidth capacity B_{uv} may be different from link (v,u) bandwidth capacity B_{vu}.

In this work, two different one hundred node random networks are generated, according to Table 1. Network 2 has a higher number of links than Network 1, according to the probability function in (20). The average bandwidth in both networks is $\overline{B} = 100$ Mbps.

6.2. DIMRO performance evaluation

The DIMRO algorithm is compared to the optimal solution of the Steiner tree problem when all link costs are equal to 1. If every link has a cost equal to 1, then 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 proposed in [15,40]. We implemented the Integer Linear Programming (ILP) problem in AMPL [19] and solved it with the CPLEX [1] solver.

DIMRO and the optimal solution of the Steiner tree problem with unitary costs are compared by using two metrics: the Rejection Rate and the Network Load. 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 network [39], we try to build a fixed number of source-rooted trees, the so-called *Number of Requested Trees* in the following. Multicast groups are sequentially randomly generated. Multicast group members (source and receivers) are randomly chosen among 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.

Table 1
Network parameters

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

To evaluate the performance of DIMRO in the different simulated scenarios, we use two metrics, namely the Rejection Rate and the Network Load.

(1) The Rejection Rate is defined as

$$\text{Rejection Rate} = \frac{\text{Number of Rejected Trees}}{\text{Number of Requested Trees}}, \tag{21}$$

where the Number of Requested Trees is the total number of source-rooted trees sequentially generated, while the Number of Rejected Trees is the number of requested source-rooted trees that cannot be built because there are not enough resources in the network.

(2) The Network load $\bar{\rho}$ is defined as

$$\bar{\rho} = \frac{\sum_{\{(u,v)\in\mathcal{E}\}}\rho_{uv}}{|\mathcal{E}|}, \tag{22}$$

where $\mathcal{E}$ is the set of network links, $|\mathcal{E}|$ its cardinality, and ρ_{uv} is the Link Load of link (u,v), defined as

$$\rho_{uv} = \frac{\text{Used Bandwidth of link } (u,v)}{\text{Bandwidth Capacity of link } (u,v)}. \tag{23}$$

Fig. 3 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 of Requested Trees is less than 2500. When the Number of Requested Trees overcomes this threshold, DIMRO shows a lowerRejection Rate.

Both DIMRO and OSTP Rejection Rates grow at the growing of the Number of Requested Trees because the same bottlenecks occur. These bottlenecks depend on the network topology and cannot be avoided, but the

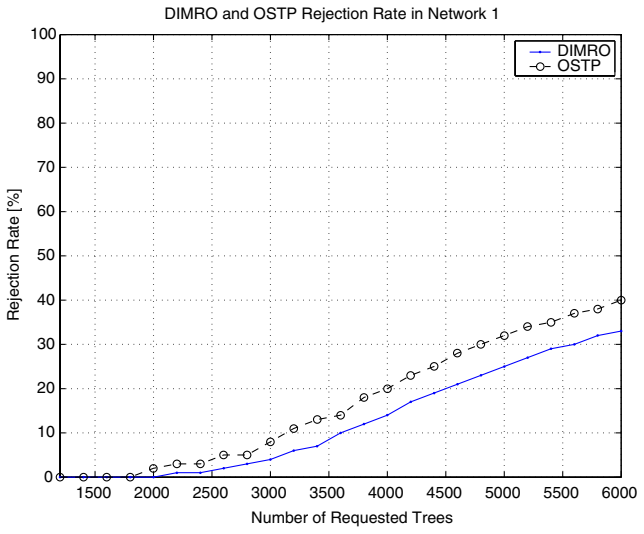


Fig. 3. DIMRO and OSTP Rejection Rate in Overlay Network 1.

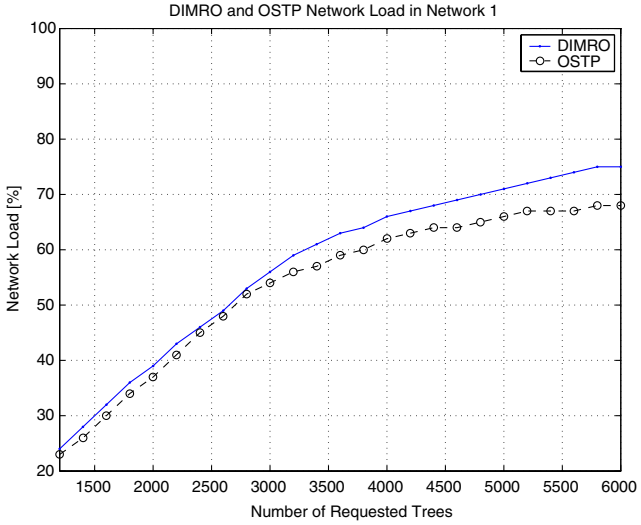


Fig. 4. DIMRO and OSTP Network Load in Overlay Network 1.

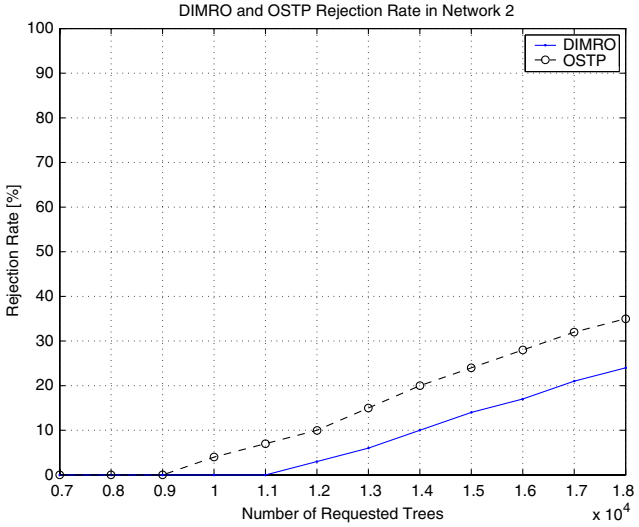


Fig. 5. DIMRO and OSTP Rejection Rate in Overlay Network 2.

DIMRO Rejection Rate is lower because bottlenecks occur later. Fig. 4 shows that the DIMRO Network Load is lightly lower than the OSTP Network Load until the Rejection Rate is the same. Then, since DIMRO rejects less trees, its Network Load is higher than the OSTP one.

Network 2 has a higher number of links than Network 1, according to (20) and the network parameters in Table 1. In this second type of network, the DIMRO Rejection Rate is significantly lower than the OSTP Rejection Rate (Fig. 5). This is because less unavoidable bottlenecks[2] exist. Since Network 2 has a higher number of links, the number of possible paths between two nodes grows, and it is easier for the DIMRO algorithm to avoid bottlenecks. A lower Network Load (Figs. 4 and 6) with a higher Rejection Rate (Figs. 3 and 5) proves that OSTP does not use efficiently network resources. In fact, bottlenecks degrade network performance and not all available resources can be exploited.

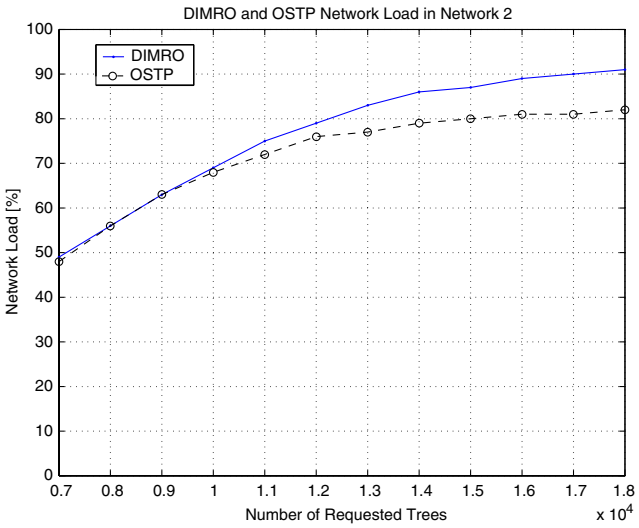


Fig. 6. DIMRO and OSTP Network Load in Overlay Network 2.

6.3. DIMRO-GS performance evaluation

To evaluate the performance of DIMRO-GS and FTM in different simulated scenarios, we use the two metrics previously introduced (slightly adapted to the group-shared case, as will be clarified in the following) plus a third metric, namely the Number of Links.

(1) The Rejection Rate is defined as in (21), but now the Number of Requested Trees is the total number of shared trees sequentially generated, and the Number of Rejected Trees is the number of requested shared trees that cannot be built because there are not enough resources in the network. Shared groups are sequentially randomly generated. For each multicast group, members are randomly chosen among network nodes. The number of members for each multicast group is uniformly distributed from 5 to 15, and the bandwidth requirement of each multicast group is uniformly distributed from 0.1 to 2 Mbps.

(2) The Network Load is defined exactly as in (22).

(3) The Number of Links is the total number of used links to connect each group member.

For each simulation several experiments have been run to ensure a small confidence region (95% relative confidence intervals smaller than 5%). Simulation results for both Network 1 and Network 2 show (Figs. 7 and 9) that at the growing of the Number of Requested Trees, the FTM Rejection Rate is significantly higher than the one of the DIMRO-GS algorithm. This can be explained by considering that the FTM algorithm uses a higher amount of network resources. In fact, when the Number of Requested Trees increases, the FTM Network Load is sig-

[2] An unavoidable bottleneck is a bottleneck that depends on the network topology and not on the algorithm used in the network. For example, if only one path exists between two nodes, all communications between these two nodes must use this path.

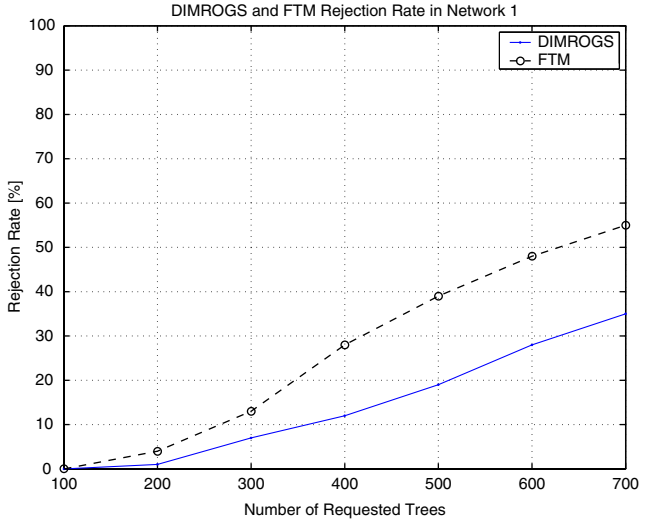


Fig. 7. DIMRO-GS and FTM Rejection Rate in Overlay Network 1.

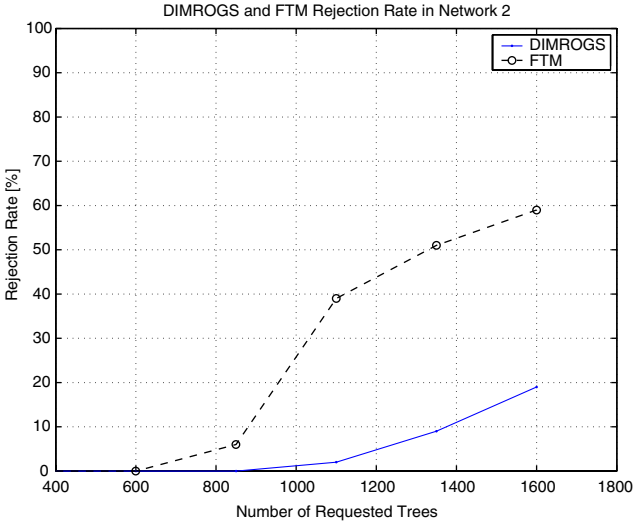


Fig. 9. DIMRO-GS and FTM Rejection Rate in Overlay Network 2.

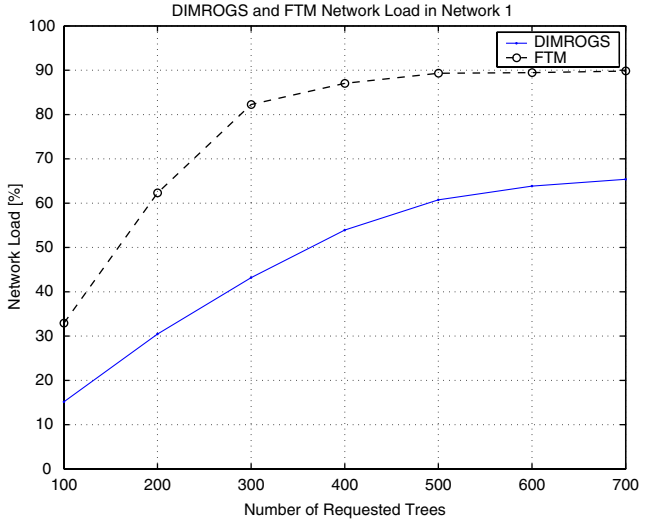


Fig. 8. DIMRO-GS and FTM Network Load in Overlay Network 1.

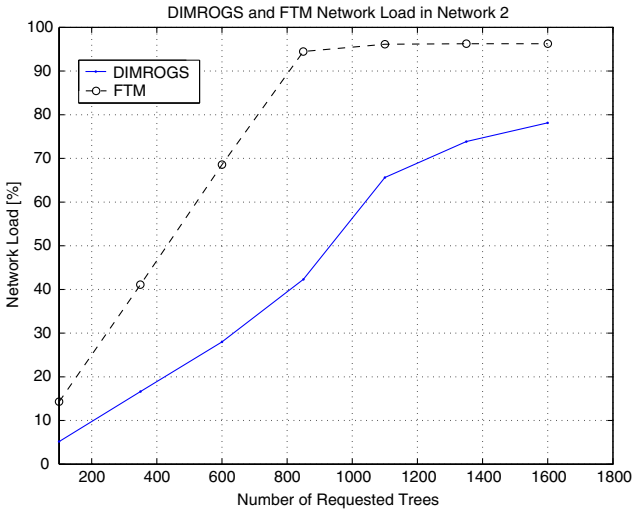


Fig. 10. DIMRO-GS and FTM Network Load in Overlay Network 2.

nificantly higher than the DIMRO-GS Network Load (Figs. 8 and 10). This implies that network resources *saturate* later when DIMRO-GS is used, causing a lower Rejection Rate for this algorithm. The lower is the number of links, the more bottlenecks degrade network performance. Figs. 9 and 10 show that DIMRO-GS achieves a lower Rejection Rate and a higher Network Load in Network 2. Conversely, in Network 1 (Figs. 7 and 8) bottlenecks do not allow to efficiently exploit all network resources.

Moreover, the Number of Links in a shared tree built by the FTM algorithm is higher than the Number of Links in the same shared tree built by DIMRO-GS, as shown in Fig. 11. This means that the amount of network resources used by FTM is higher than the amount of network resources exploited by DIMRO-GS. At the growing of the group size, the Number of Links that make up the shared tree built by the proposed algorithm grows slower

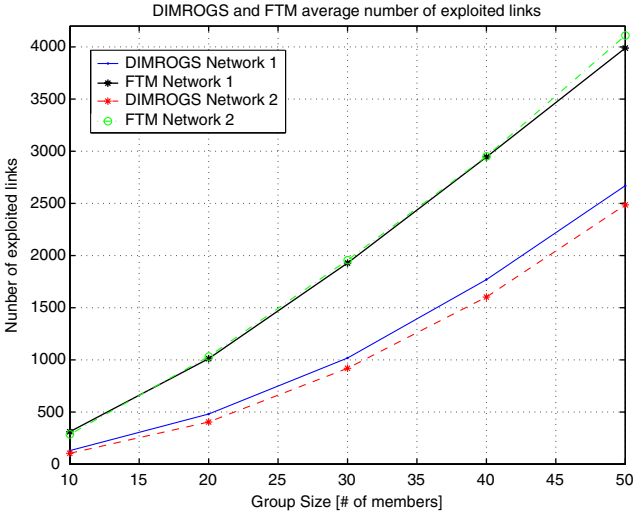


Fig. 11. DIMRO-GS and FTM average number of link.

than the Number of Links that makes up the shared tree built by the FTM algorithm, in both Network 1 and Network 2 (Fig. 11).

Fig. 11 shows also that the Number of Links that make up the shared tree built by the DIMRO-GS algorithm is greater in the first network than in the second one. This is because when the number of links in the network grows, the value of the exponent decreases. Thus, paths built by the proposed algorithm are shorter, and the Number of Links that makes up the shared tree is lower.

6.4. DIMRO and DIMRO-GS performance evaluation in differentiated service overlay networks

In this section, we compare the performance of the extended DIMRO and DIMRO-GS algorithms in QoS-aware overlay networks with the simple case in which a group member/receiver cannot reuse the bandwidth exploited by another group member/receiver with more stringent QoS requirements.

Network nodes are randomly placed on a Cartesian coordinate grid and links are generated according to the probability function in (20). Since we need to generate a QoS-aware overlay network, we consider the bandwidth capacity of a link for each service class. For the sake of simplicity, let us consider a differentiated service overlay network with only four service classes, which we indicate with A, B, C, D, where A is the highest service class and D is the lowest one. Let $B_{uv}(\mathrm{cl})$ be the bandwidth capacity of overlay link (u,v) for service class cl, $\mathrm{cl} \in \{A, B, C, D\}$. The bandwidth capacity $B_{uv}(\mathrm{cl})$ for each service class on link (u,v) is randomly generated by using an uniform distribution with mean bandwidth $\overline{B}(\mathrm{cl})$. For this simulation campaign, a one hundred node network has been randomly generated, with parameters $\alpha = 0.2$ and $\beta = 0.4$, i.e., Network 1 according to Table 1. The link bandwidth capacity for each service class $\mathrm{cl} \in \{A, B, C, D\}$ is uniformly distributed with mean bandwidth $\overline{B}(\mathrm{cl})$ equal to 25 Mbps.

We consider the Network Load as defined in (22) for each service class, when 50 randomly generated source-rooted trees are built by the DIMRO algorithm. Each receiver asks for a rate uniformly distributed in the range $[0.1, 2]$ Mbps, and a service class randomly selected within the set $\{A, B, C, D\}$. Moreover, a 5-channel layering scheme is used, as described in Section 4.1.

For each simulation several experiments have been run to ensure a small confidence region (95% relative confidence intervals smaller than 5%). Fig. 12 shows that DIMRO in differentiated service overlay networks performs better than its not extended counterpart. Furthermore, the Network Load decreases with the service class. In particular, the lowest service class D has the lowest *Network Load* because D class receivers can reuse bandwidth already exploited by all other receivers. In differentiated service overlay networks DIMRO performs better than in non QoS-aware networks, and this becomes more and more evident when the number of receivers increases.

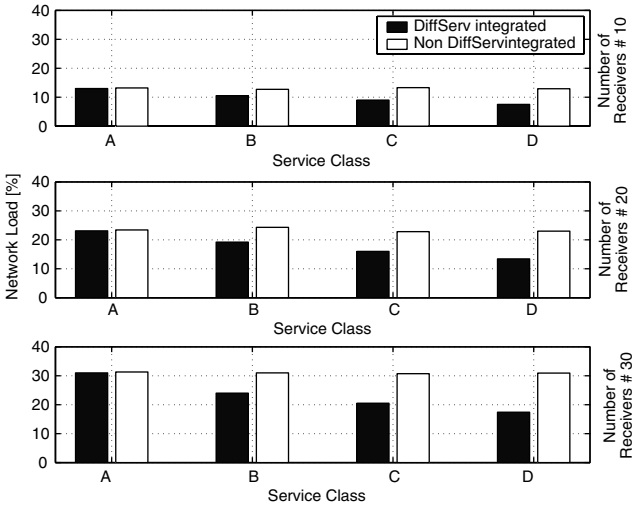


Fig. 12. DIMRO Network Load vs. service class.

Fig. 12 shows that for 30 receivers the Network Load for service classes B, C and D is significantly lower then the Network Load relative to these service classes when DIMRO works in non-QoS-aware overlay networks (the Network Load is reduced respectively by 23%, 33%, and 40%, as can be easily seen in Fig. 12).

As far as DIMRO-GS is concerned, we consider the Network Load of each service class when a multicast group is randomly generated and the DIMRO-GS algorithm is used to build the corresponding shared tree. The group asks for a rate uniformly distributed in the range $[0.1, 2]$ Mbps, and each group member asks for a service class randomly selected within the set $\{A, B, C, D\}$.

Fig. 13 shows that DIMRO-GS in QoS-aware overlay networks performs better than in non-QoS-aware overlay networks. The Network Load decreases with the service class (the lowest service class D has the lowest Network Load). Moreover, performance of DIMRO-GS in QoS-

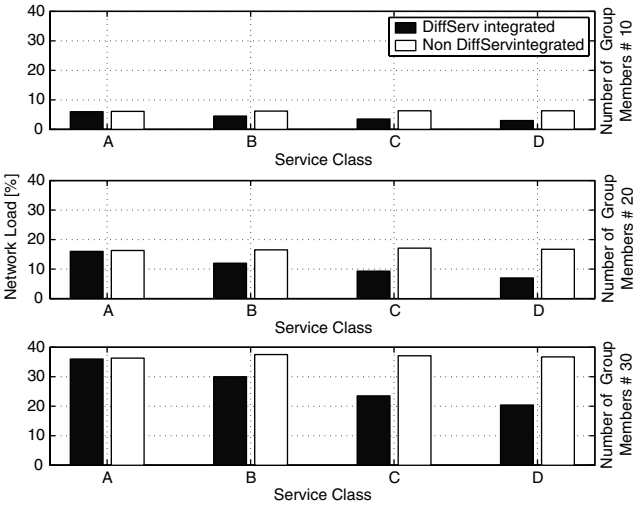


Fig. 13. DIMRO-GS Network Load vs. service class.

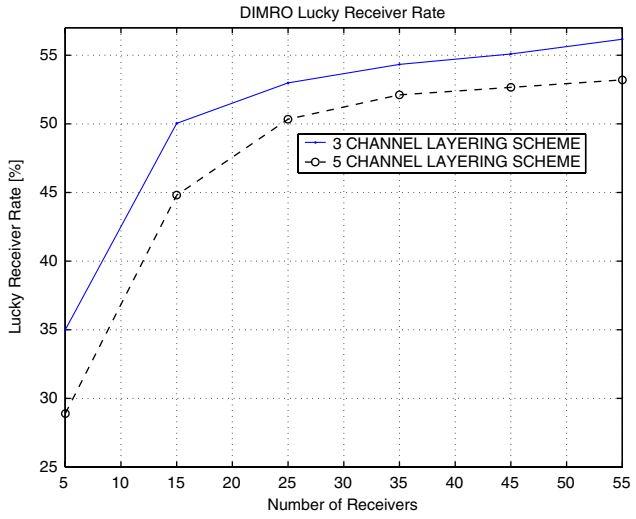


Fig. 14. DIMRO Lucky Receivers.

aware overlay networks improves when the number of group members increases.

When a group member/receiver reuses resources already exploited by a receiver with more stringent QoS requirements, it obtains a better QoS than that it requires, with no extra cost to the network. We call *lucky receivers* (*lucky members* in the DIMRO-GS case) those receivers (members) that use the bandwidth relative to a higher class at least in one link.

Figs. 14 and 15 show that the *lucky receiver (member) rate*, i.e., the percentage of receivers (members) that receive a better quality of service than that they requested, grows as the multicast group enlarges. Fig. 14 also shows that DIMRO *lucky receiver rate* grows at the decreasing of the number of channels, i.e., the number of cumulative rates, in the layering scheme. In fact, by reducing the number of possible cumulative rates, it becomes more likely that receivers with different QoS requirements ask for the same rate and, thus, it is more likely that receivers asking for less stringent QoS requirements can reuse resources already exploited by receivers with more stringent QoS requirements. In other words, by reducing the number of channels, less receivers can obtain the desired rate, but more receivers attain a better QoS, which implies less losses, less retransmissions, and an increased average throughput.

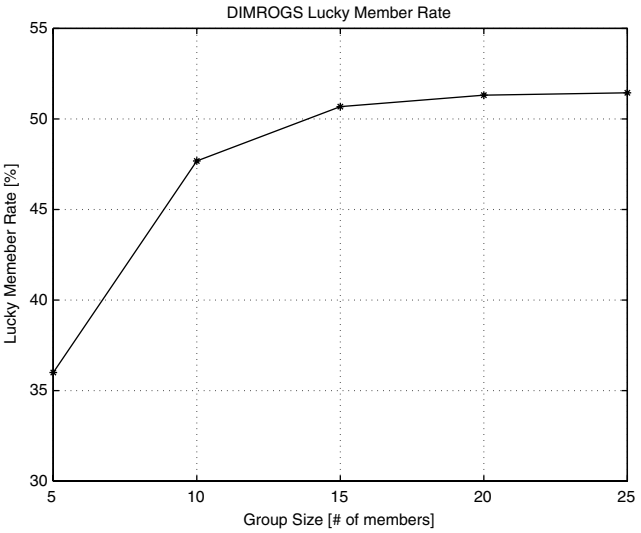


Fig. 15. DIMRO-GS Lucky Members.

7. Conclusions and future research

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 in such a way as to avoid traffic congestion and fluctuation in the underlay network, which cause low 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. Our future research will focus on dynamic multicast groups on overlay networks, and on the dynamic interactions between overlay and underlay networks.

Acknowledgement

The authors are in debted to the anonymous reviewers whose unselfish comments greatly improved this work.

Appendix A. Channel rate assignment algorithm

In this Appendix, we present an efficient algorithm to exactly solve problem **P**, presented in Section 4.1, in polynomial time. An example of the algorithm in a simplified scenario will be given after its formal description (Figs. 16–20). The algorithm goes through two stages:

(1) We build a directed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, where nodes of the graph are the N different requested rates together with an exit node n_{exit}, i.e., $\mathcal{V}=\{W_1,..,W_N\}\cup\{n_{\text{exit}}\}$, thus $|\mathcal{V}|=N+1$;

(2) We solve a *maximum cost path problem* from node W_1 to node n_{exit} in the constructed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$. At this point, all the nodes belonging to this path will be those

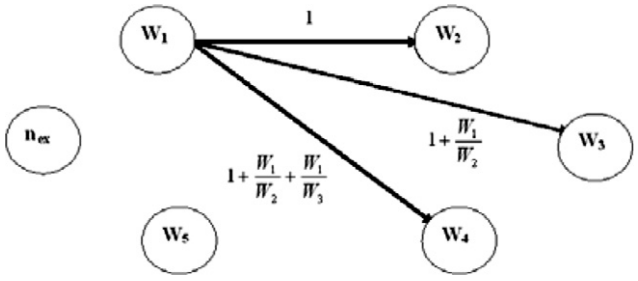


Fig. 16. Determining channel rates: *Step* 1.

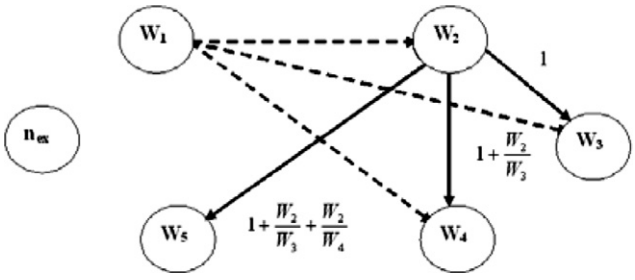


Fig. 17. Determining channel rates: *Step* 2.

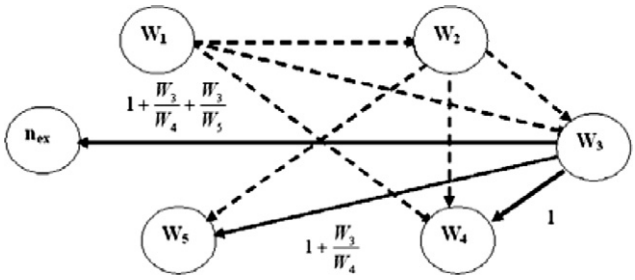


Fig. 18. Determining channel rates: *Step* 3.

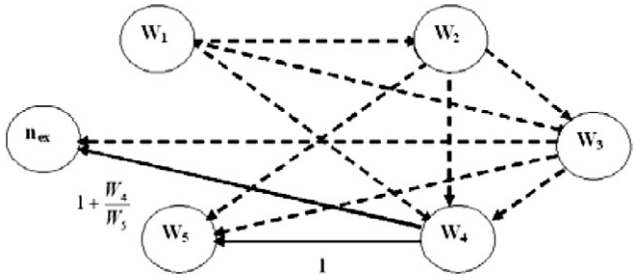


Fig. 19. Determining channel rates: *Step* 4.

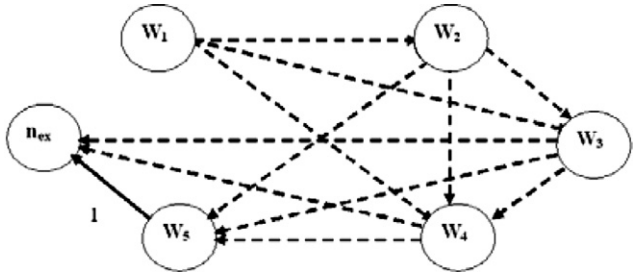


Fig. 20. Determining channel rates: *Step* 5.

cumulative channel rates $L_1, \ldots, L_K^*$ that form an optimal solution of problem **P**.

Let us now assume $L_1 = \min\{L_1, \ldots, L_{K^*}\} = W_1$. If $L_1 > W_1$, since $L_{K^*} > \ldots > L_1 > W_1$ from (9), then receivers that asked for rate W_1 could not subscribe any channel without incurring in data loss. In fact, if they subscribed channel $w_1 = L_1 > W_1$, they would be overwhelmed by arriving packets. Thus, this means that it must necessarily be $L_1 = W_1$, as initially assumed. The proposed algorithm to exactly solve **P**, proceeds into steps, hereafter described.

Step 1: Starting from node (rate) W_1, a link $(1, j)$ from W_1 to W_j $(\forall j = 2,\ldots, K^* + 1)$ is added to the graph. Since there are K^* channels, the set of all possible values for L_2 is $\{W_2, \ldots, W_{K^*+1}\}$, thus at least $K^* - 2$ values $\{W_{K^*+2}, \ldots, W_N\}$ remain for the others $K^* - 2$ cumulative rates $L_3, \ldots, L_{K^*}$.

The cost $c(1,j)$ of each link $(1,j)$ $(\forall j = 2,\ldots, K^* + 1)$ is the partial value of the objective function of **P**, determined by the choice $L_1 = W_1$ for all those receivers that ask for rates from W_1 to W_{j-1}. Thus, the cost of link $(1,j)$ is defined as,

$$c(1,j) = \sum_{n=1}^{j-1} \eta_n \cdot \frac{W_1}{W_n}. \tag{24}$$

Step i, $i = 2,\ldots, K^* - 1$: A link (i,j) from W_i to W_j $(\forall j = i + 1,\ldots, K^* + i)$ is added to the graph. The set $\{W_{i+1}, \ldots, W_{K^*+i}\}$ is determined by considering that at this step, values of the cumulative rates, $L_1 = W_1,\ldots, L_i = W_i$, could have already been assigned (it is only an admissible solution, not necessarily the optimal one). In this limit case, at least $K^* - i - 1$ values $\{W_{K^*+i+1}, \ldots, W_N\}$ must remain for the others $K^* - i - 1$ values of the cumulative rate $L_{i+2}, \ldots, L_{K^*}$.

The cost $c(i, j)$ of the link (i,j) is defined as,

$$c(i,j) = \sum_{n=i}^{j-1} \eta_n \cdot \frac{W_i}{W_n}. \tag{25}$$

The cost $c(i,j)$ is the partial value of the objective function of **P**, determined by the choice of the cumulative rate W_i for all those receivers that ask for rates from W_i to W_{j-1}.

Step i, $i = K^*,\ldots, N - 1$: A link (i,j) from W_i to W_j $(\forall j = i + 1,\ldots, N)$ is added to the graph. The cost $c(i, j)$ of link (i,j) is defined according to (25). Furthermore, a link from node (rate) W_i to the exit node n_{exit} is added to the graph. In fact, at *Step i*, $i = K^*$, all K^* values of the cumulative rates $L_1, \ldots, L_{K^*}$ could have already been assigned. If there are no more cumulative rates, i.e., no more channels, then the cumulative rate $L_{K^*} = W_i$ is assigned to all the receivers that ask for rates from W_i to W_N.

The cost $c(i, n_{\text{exit}})$ of the link (i, n_{exit}) is defined as,

$$c(i, n_{exit}) = \sum_{n=i}^{N} \eta_n \cdot \frac{W_i}{W_n}. \tag{26}$$

Eq. (26) represents the partial value of the objective function of **P**, determined by the choice of the cumulative rate W_i for all the receivers that ask for rates form W_i to W_N.

Step N: Finally, only one link (N, n_{exit}) from node (rate) W_N to the exit node n_{exit} is added to the graph. The cost $c(N, n_{\text{exit}})$ of the link (N, n_{exit}) is simply $c(N, n_{\text{exit}}) = \eta_N$, which is the contribution to the objective function of the choice $L_{K^*} = W_N$.

Example. Let us consider $N = 5$ different requested rates $W_1 < W_2 < \ldots < W_5$ and let us suppose that only $K^* = 3$

channels are available. Let us assume, for the sake of clarity, that $\eta_n = 1$, $n = 1, \ldots, 5$.

Step 1: L_1 is set to W_1 and links from node W_1 to nodes W_2, W_3 and W_4 are added to the graph (Fig. 16). There is no link from W_1 to W_5 because the possible cumulative rates are three ($K^* = 3$) and it cannot be $L_2 = W_5$. The cost of each link (W_1, W_j) is the contribution of the choice $L_2 = W_j$ to the objective function of **P**.

Step 2: Links from node W_2 to nodes W_3, W_4, and W_5 are added to the graph (Fig. 17). The cost of each link (W_2, W_j) is the contribution of the choice $L_3 = W_j$.

Step 3: Links from node W_3 to nodes W_4, W_5, and n_{exit}, are added to the graph (Fig. 18). The cost of each link (W_3, W_j) is the contribution of the choice $L_3 = W_j$ to the objective function when $L_2 = W_3$, while the cost of the link (W_3, n_{exit}) is the contribution of the choice $L_3 = W_3$ when $L_2 = W_2$.

Step 4: Links from node W_4 to nodes W_5 and from node W_4 to n_{exit} are added to the graph (Fig. 19). The cost of link (W_4, W_5) is the contribution of the choice $L_3 = W_5$ to the objective function when $L_2 = W_4$, while the cost of the link (W_4, n_{exit}) is the contribution of the choice $L_3 = W_4$ to when $L_2 = W_2$ or $L_2 = W_3$.

Step 5: Only a link from node W_5 to n_{exit} is added to the graph (Fig. 20). The cost of the link (W_5, n_{exit}) is the contribution of the choice $L_3 = W_5$ to the objective function of **P** when $L_2 = W_2$, $L_2 = W_3$ or $L_2 = W_4$.

Cumulative channel rates $L_1, \ldots, L_{K^*}$ that form an optimal solution of problem **P** can be found by solving a maximum cost path problem in the constructed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

Let Ω be the set of all possible values of the objective function $\Phi(L_1, \ldots, L_{K^*})$ of **P** on the admissible region. Let Π be the set of all possible paths from node W_1 to n_{exit} in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with maximum number of hops equal to K^*, and let D_Π be the set of the lengths of paths in the set Π, i.e., the set of the costs of paths in the set Π. The graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ has been built in a way that the set D_Π coincides with the set Ω ($D_\Pi \equiv \Omega$). Thus, to find the optimal solution of **P**, it is sufficient to find the longest path from node W_1 to n_{exit} in the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

Since the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is acyclic [22], the Bellman–Ford algorithm [14] can be used to find the longest path from node W_1 to n_{exit} with a number of hop lower or equal to K^*. Nodes of this path but n_{exit} represent the set of optimum cumulative rates $L_1, \ldots, L_{K^*}$, i.e., the optimal solution of **P**.

Computational complexity: The graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ has a number of nodes $|\mathcal{V}| = N + 1$ and a number of links equal to

$$|\mathcal{E}| = (K^* - 1) \cdot (N - K^* - 1) + \sum_{i=K^*}^{N} (1 + N - i). \tag{27}$$

The Bellman–Ford algorithm, which dominates most of the computing time of the algorithm, has a computational complexity $O(|\mathcal{V}| \cdot |\mathcal{E}|)$. Thus, the complexity of the proposed algorithm for the exact determination of logical channel rates is $O(K^* \cdot N^2)$.

References

[1] CPLEX, <http://www.cplex.com/>.

[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, Englewoof Cliffs, NJ, 1993.

[3] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, R. Morris, Resilient overlay networks, in: Proceedings of ACM Symposium on Operating Systems Principles (SOSP), October 2001, Banff, Canada, 2001.

[4] T. Ballardie, Core Based Trees (CBT version 2) Multicast Routing, Technical report, IETF RFC 2189, September 1997.

[5] T. Ballardie, P. Francis, J. Crowcroft, Core based trees (CBT): an architecture for scalable inter-domain multicast routing, Computer Communication Review 23 (Oct.) (1993) 85–95.

[6] T. Ballardie, P. Francis, J. Crowcroft, Core based trees (CBT) and architecture for scalable inter-domain multicast routing, in: Proceedings of ACM Special Interest Group on Data Communications (SIGCOMM), September 1993, San Francisco, USA, 1993.

[7] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, S. Khuller, Construction of an efficient overlay multicast infrastructure for realtime applications, in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), March 2003, vol. 2, San Francisco, CA, USA, 2003, pp. 1521–1531.

[8] S. Baset, H. Schulzrinne, An analysis of the skype peer-to-peer internet telephony protocol, in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), April 2006, Barcelona, Spain, 2006.

[9] S. Bhattacharyya, J. Kurose, D. Towsley, Efficient multicast flow control using multiple multicast groups, Technical report, CMPSCI, TR 97-15, March 1997.

[10] Y. Birk and D. Crupnicoff, A multicast transmission schedule for scalable multi-rate distribution of bulk data using non-scalable erasure-correcting codes, in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), July 2003, San Francisco, USA, 2003.

[11] K. Calvert, M. Doar, E. Zegura, Modeling Internet topology, IEEE Communications Magazine 35 (6) (1997) 160–163.

[12] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, A. Singh, Splitstream: high-bandwidth multicast in cooperative environments, in: Proceedings of ACM Symposium on Operating Systems Principles (SOSP), October 2003, Bolton Landing, NY, USA, 2003.

[13] M. Castro, P. Druschel, A. Kermarrec, A. Rowstron, SCRIBE: a large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Selected Areas in Communications 20 (8) (2002) 1489–1499.

[14] C. Cheng, R. Riley, S.P.R. Kumar, J.J. Garcia-Luna-Aceves, A loop-free Bellman–Ford Routing protocol without bouncing effect, in: Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), September 1989, Oakland, California, USA, 1989, pp. 224–237.

[15] A. Claus, N. Maculan, Une Nouvelle Formulation du Probleme de Steiner sur un Graphe. Technical report, Centre de Recerche sur les Transports, Universite de Montreal, 1983.

[16] Z. Duan, Z. Zhang, Y. Hou, Service overlay networks: SLAs, QoS, and bandwidth provisioning, IEEE/ACM Transactions on Networking 11 (6) (2003) 870–883.

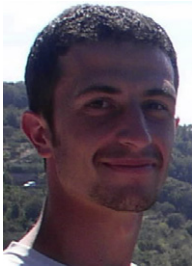
[17] J. Fan and M. Ammar, Dynamic topology configuration in service overlay networks: a study of reconfiguration policies, in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), April 2006, Barcelona, Spain, 2006.

[18] N. Feamster, D. Andersen, H. Balakrishnan, F. Kaashoek, Measuring the effects of internet path faults on reactive routing, in: Proceedings of ACM International Conference on Measurement

and Modeling of Computer Systems (SIGMETRICS), June 2003, San Diego, CA, USA, 2003.
[19] R. Fourer, D.M. Gay, B.W. Kernighan, AMPL: a modeling language for mathematical programming, Duxbury Press, Cole Publishing Co., 2002.
[20] P.G. Franciosa, D. Frigioni, R. Giaccio, Semi-dynamic shortest paths and breadth-first search on digraphs, in: Proceedings of Symposium on Theoretical Aspects of Computer Science, Lubeck, Germany, February/March 1997, pp. 33–46.
[21] M.R. Garey, D.S. Johnson, Computer and intractability: a guide to the theory of NP-completeness, W.H. Freeman and Co., San Francisco, CA, 1979.
[22] J.L. Gross, J. Yellen, Handbook of graph theory, Discrete Mathematics and Its Applications, vol. 25, CRC Press, Boca Raton, 2003.
[23] X. Gu, K. Nahrstedt, R. Chang, C. Ward, QoS-assured service composition in managed service overlay networks, in: Proceedings IEEE International Conference on Distributed Computing Systems (ICDCS), May 2003, Providence, Rhode Island, USA, 2003.
[24] T. Jiang, M.H. Ammar, E.W. Zegura, Inter-receiver fairness: A novel performance measure for multicast ABR sessions, in: Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Madison, Wisconsin, USA, 1998.
[25] D. Kostic, A. Rodriguez, J. Albrecht, A. Vahdat, Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh, in: Proceedings of ACM Symposium on Operating Systems Principles (SOSP), October 2003, Bolton Landing, NY, USA, 2003.
[26] L. Lao, J.-H. Cui, M. Gerla, TOMA: a viable solution for largescale multicast service support, in: Proceedings of IFIP International Conference of Networking, May 2005, Waterloo, Ontario, Canada, 2005, pp. 906–917.
[27] Z. Li, P. Mohapatra, Qron: Qos-aware routing in overlay networks, IEEE Journal on Selected Areas in Communications (JSAC) 22 (1) (2004) 29–40.
[28] D. Lichtenstein, Planar formuale and their use, SIAM Journal on Computing 11 (2) (1982) 329–343.
[29] C.P. Low, N. Wang, On finding feasible solutions to group multicast routing problem, IEICE Transactions on Communications E85-B (1) (2002) 268–277.
[30] A. Nakao, L. Peterson, A. Bavier, A routing underlay for overlay networks, in: Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), June 2003, San Diego, CA, USA, 2003.
[31] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, A. Mohr, Chainsaw: eliminating trees from overlay multicast, in: Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS), February 2005, Ithaca, NY, USA, Feb. 2005.
[32] P. Paul and S. Raghavan, Survey of multicast routing algorithms and protocols, in: Proceedings of the International Conference on Computer Communication (ICCC), August 2002, Bandra, Mumbai, India, 2002.
[33] L.H. Sahasrabuddhe, B. Mukherjee, Multicast routing algorithms and protocols: a tutorial, IEEE Network 14 (1) (2000) 90–102.
[34] S.Y. Shi, J.S. Turner, Routing in Overlay Multicast Networks, in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), June 2002, New York, NY, USA, 2002.
[35] S.Y. Shi, J.S. Turner, Routing in Overlay Multicast Networks, in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), June 2002, vol. 3, New York, NY, USA, 2002. pp. 1200–1208.
[36] H. Takahashi, A. Matsuyama, An approximate solution for the steiner problem in graphs, Mathematica Japonica 6 (1980) 573–577.
[37] S. Vieira, J. Liebeherr, Topology design for service overlay networks with bandwidth guarantees, in: Proceedings of IEEE International Workshop on Quality of Service (IWQoS), June 2004, Montreal, Canada, 2004, pp. 211–220.
[38] N. Wang, C.P. Low, On finding feasible solutions to group multicast routing problem, in: Proceedings of IFIP International Conference of Networking, May 2000, Paris, France, 2000.
[39] B.M. Waxman, Routing of multipoint connections, IEEE Journal on Selected Areas in Communications 6 (9) (1988) 1617–1622.
[40] R.T. Wong, A dual ascent approach for steiner tree problems on a directed graph, Mathematical Programming 28 (1984) 271–287.
[41] H. Zhang, J. Kurose, D. Towsley, Can an overlay compensate for a careless underlay? in: Proceedings of IEEE Conference on Computer Communications (INFOCOM), April 2006, Barcelona, Spain, 2006.
[42] Y. Zhu, C. Dovrolis, M. Ammar, Dynamic overlay routing based on available bandwidth estimation: a simulation study, Computer Networks (Elsevier) 50 (6) (2006) 742–762.

Dario Pompili graduated in Telecommunications Engineering (summa cum laude) from the University of Rome 'La Sapienza', Italy, in 2001. In 2004, he earned from the same university the Ph.D. Degree in System Engineering. In 2003, he worked on sensor networks at the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, as a Visiting Researcher. In 2007, he earned the Ph.D. Degree in Electrical and Computer Engineering at the Georgia Institute of Technology under the guidance of Dr. I.F. Akyildiz. Starting August 2007, he is an Assistant Professor in the Electrical and Computer Engineering Department at Rutgers, the State University of New Jersey. His main research interests are in wireless ad hoc and sensor networks, underwater acoustic sensor networks, and overlay networks.

Caterina Scoglio graduated in Electrical Engineering (summa cum laude) from the University of Rome 'La Sapienza', Italy, in 1987. In 1988, she received a Post-graduate Degree in Mathematical Theory and Methods for System Analysis and Control from the same university. From 1987 to 2000, she was with Fondazione Ugo Bordoni, Rome, as a Research Scientist. From 2000 to 2005, she was with the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, as a Research Engineer. Currently, she is an Associate Professor in the Electrical and Computer Engineering Department at Kansas State University. Her research interests include optimal design and management of overlay networks.



Luca Lopez graduated in Electronic Engineering (summa cum laude) from the University of Rome 'La Sapienza', Italy, in 2003, with a thesis titled "Resource Optimization for Internet Multicast Applications". His main research interests include multicast communications, QoS in overlay networks, and digital signal processing.