Telecommun Syst (2014) 56:121–139
DOI 10.1007/s11235-013-9823-x

Modeling and optimization of maximum flow survivable overlay multicast with predefined routing trees

Michał Kucharzak · Krzysztof Walkowiak

Published online: 30 August 2013

Abstract Overlay networks employ underlying network technologies in order to provide end-system related communication and over the years overlays have been getting more and more attention in research community and in business world as well. Since overlays tackle many drawbacks present in pure "link-router-network" engineering, they have become an excellent solution for multimedia-oriented applications. A good example comprises multicast communications, where an overlay system, in contrast to IP Multicast, eliminates many issues related to scalability or management control. This paper focuses on modeling and optimization of overlay multicast networks aimed at realizing maximum throughput with survivability constraints, where survivability defines the ability of a multicast system to limit potential throughput losses in case of a failure of single virtual link. We present linear formulation derived from fractional tree packing problems based on predefined topologies which may route multicast traffic. Linear model might be used for obtaining optimal multicast structures, however its applicability is limited by increasing sizes of networks. Hence, we also design and evaluate heuristic searches dedicated to optimization of maximum flow survivable overlay multicast networks.

Keywords Overlay multicast · Survivability · Optimization

M. Kucharzak (✉) · K. Walkowiak
Department of Systems and Computer Networks, Wroclaw University of Technology, Wroclaw, Poland
e-mail: michal.kucharzak@pwr.wroc.pl

K. Walkowiak
e-mail: krzysztof.walkowiak@pwr.wroc.pl

1 Introduction

Multicast system defined for overlay networks uses routing implemented by end hosts and has already outperformed IP Multicast during recent years. It derives from the fact that the overlay architecture for multicast streaming reduces many pure network layer drawbacks and provides potential scalability and easy deployment of new protocols independent of the network layer solutions and service providers interfaces at relatively low costs. Over the years, a lot of research have been devoted to support overlay multicast and many protocols for such concepts have been already proposed [3, 5, 27, 28, 37, 38, 46, 49].

Since most of the overlay multicast protocols focus on providing end-to-end content distribution at given streaming rate, a significant portion of works on optimization of overlay multicast are devoted to satisfy minimum flow costs or delays and guarantee streaming on given data rates (e.g. [23, 39, 44]). On the other hand, the objective of content distribution might also consider maximization of system's throughput. For example, in [45] the authors employ conceptual flows and formulate linear program with node capacity constraints for maximizing data rate allocation (DRA) and even though the linear model formulates optimal throughput that might be achieved globally in the system, it has been already shown to be impractical in overlay applications with large scale networks. To deal with this issue, our previous works proposed slightly different idea which is based on predefined trees selection [25, 26]. Note that, such a concept is rather a kind of relaxation of the problem in [45] and the results provided by our linear approach are not guaranteed to reach the same optimality level as the DRA, it has been proved to be an excellent trade-off between result quality and operational time that is crucial in either real-time or large scale multicast applications. Additionally, in [24] we consider an

overlay system that provides survivability against a single virtual link failure and guarantee limited losses of throughput in overlay multicast network.

In this paper, we extend our previous work [24] presented at 3rd International Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) Congress, held in Budapest, Hungary on October 5–7, 2011. The paper devoted to survivable maximum flow routing in multicast overlays comprised linear models comparison and now, we discuss maximum flow survivable overlay multicast with predefined routing trees where novelty and main contributions include design, implementation and evaluation of heuristic and metaheuristic strategies. Since the applicability of linear-based FST approach might be hindered due to general reasons in cases the problem instances grow fast in their size, we propose two types of algorithms: a constructive search—remaining flow selection strategy—which is based on actual utility of available routing on predefined trees and three basic concepts with random-based decisions: random search, simulated annealing and hybrid search. We compare the effectiveness of the algorithms in the meanings of result qualities and required computational time as well. In order to find optimal solutions, if possible, we implement the FST linear model in CPLEX solver [18].

The rest of the paper is organized as follows. Next, in Sect. 2 we illustrate fundamental differences between two basic multicast approaches: network layer located and pure overlay. In Sect. 3, we briefly refer to state of the art in modeling of overlay multicast networks with maximum flows and survivability issues that might be applied to such systems. Section 4 embraces linear models based on fractional spanning tree packing with various interpretations of survivability devoted to limit potential losses of multicast throughput while a single link failure appears. Three different interpretations of survivability are presented. First, disjointed spanning tree constraints limits the utilization of trees with common overlay arcs or edges. Second, overlay links are virtually capacitated in order to balance flows and limit their usage. And finally, relative survivability factor is introduced. The factor bounds overlay arcs or edges traffic in accordance to already achieved multicast flows. The formulations comprise polynomial number of variable and constraints as well. In Sect. 6, we focus on two latter survivability versions and various optimization algorithms are presented. We start with random search algorithm which randomly allocates flows, next, simulated annealing which improves pure random search is described. Then we describe remaining flow selection strategy that uses no random-driven decisions in its operations. Finally, hybrid search is shown. The algorithm uses both remaining flow selection strategy and local search performed in a random way. All search techniques are accompanied with their pseudocodes, we also present pseudocodes of algorithms which are crucial subsystems of the main search approaches.

2 Overlay multicast system

Multicast network technique delivers information to a group of destinations simultaneously and two basic multicast approaches might be distinguished. Network layer multicast is realized by routers inside the network: data packets are replicated at routers (Fig. 1). The mostly developed and well discussed approach for network level multicast was proposed by Deering and Cheriton about two decades ago [11] and is known as IP Multicast. This architecture implements multicast functionality in the IP layer, and from the network flows view, it is the most efficient way to perform group data distribution because packet replication is reduced to the minimum. However, deployment of IP Multicast is limited and hindered due to several sorts of reasons [13] including addressing schemes, scalability or dynamics of the routing architecture and finally, IP-based multicast development is far from being globally and widely deployed.

Alternative multicast technique slightly revolutionizes multicast-based communication and tackles some problems of network-aware IP Multicast by expanding end-system multicast [9, 10, 36, 47] and using overlay based systems

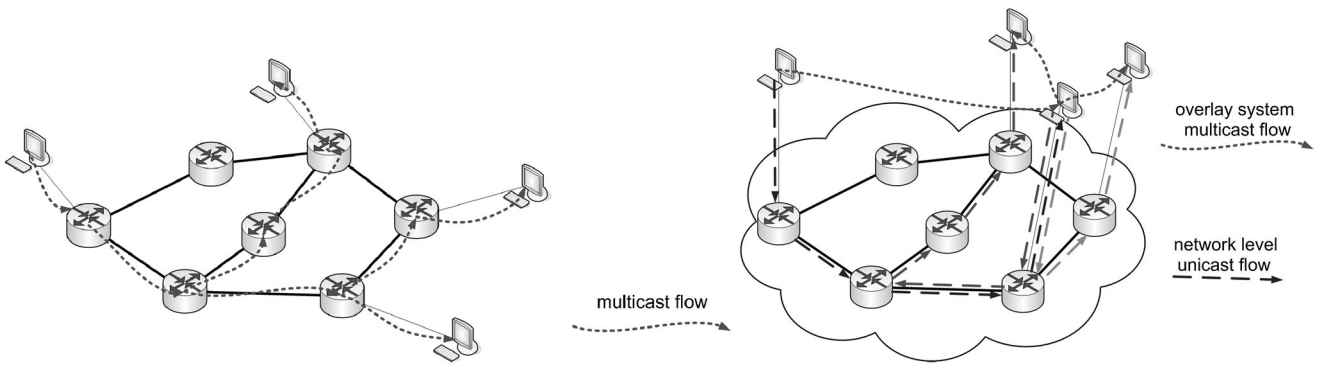


Fig. 1 Multicast flow visualization example in IP Multicast network-aware system (*left*) and overlay based multicast (*right*)

has become popular approach for multicast. In general, instead of IP routers, participating users actively contribute their upload bandwidth capacities to serve other peers in the same streaming session by forwarding their available content. Peers form and maintain a logical overlay multicast structure for efficient data transmission. Physical realization of the logical links is performed as sets of directed unicasts, mostly using well-defined IP unicast communication. Since the overlay multicast utilizes the same network link multiple times, it is less efficient than IP Multicast in terms of network redundancy. But eventually, overlay multicast has been outperforming IP Multicast during recent years.

3 Related works

In the beginning, it is worth investigating the general track of modeling of maximum flows in overlay multicast networks. We would refer to Grötschel et al. [17] and the problem of packing Steiner trees. Although the problem is well justified in graphtheoretic meaning, it does not really fit flow maximization in overlay multicast. First of all, the authors focus on finding a given number of Steiner trees in a graph with limited edge capacities, hence they try to pack some trees instead of packing as much trees as possible (maximization). Moreover, capacity conditions limit the number of trees using the same edge and no other flows than logical $\{0, 1\}$ might be assigned to the trees. Even though we could reformulate the problem in order to find maximum possible number of such trees, overlays comprising only end-hosts require rather spanning trees which are just a special case of Steiner tree.

Contrary to the packing Steiner tree, where trees are assumed to carry binary flows, in [4] Barahona already discusses the problem of packing spanning trees, where each spanning tree is assigned with non-negative weight, so that the sum of the weights over all trees is maximized and capacity of any edge cannot be exceeded by the sum of the weights of the trees containing it. However, the weights might represent flows (and also might satisfy aspects of network coding), the problem takes into account edge capacity constraints instead of node capacity constraints. Moreover, the size of the basic problem strongly depends on the number of spanning trees to be assigned, and according to the Cayley's formula [7], number of spanning trees in a given graph can reach V^{V-2} for a complete graph with V vertexes.

In accordance to aforementioned works, Garg et al. [16] formulate bandwidth maximization problems in multicast streaming data. Authors of [16] refer to end-host based system in which users are connected via network with routers, where end-host may replicate the stream while routers only forward the data. This concept may be applied to multilayered network rather than pure overlays. The problems concern unsplittable and splittable cases. The former is defined as whether it is possible to construct a multicast tree of bandwidth not less than the given value. Such a problem is proved to be NP-complete. Splittable version of the problem allows the multicast stream to be divided into substreams, and its formulation is derived from packing Steiner trees proposed in [20]. Jain et al. [20] define their objective as finding the maximum number of edge-disjoint Steiner trees in a given graph and also present a fractional version of the problem. However, both paper assume edge capacity constrains on undirected links.

Approaching the theorem of information flows, Ahlswede et al. [1], Koetter et al. [22] and Li et al. [29] show that network coding may resolve conflicts in case some flows directed to various receivers share links in the network. Consequently, an achievable multicast rate for the entire multicast transmission is limited to the data rate that might be accomplished from the source to every end-user independently. Since maximum unicast flows can be found in polynomial time (see, e.g. [2]), maximum multicast flows can be achieved by applying max unicast flow algorithms per each destination. Following the conclusion, in [30] authors formulate a linear model of throughput maximization in multicast networks. The formulation uses Kirchoff-based flow conservation constraints and undirected links with edge capacities for which upload and download exploit the same link resources simultaneously. Although, from the overlay system perspective end-hosts are usually connected via DSL-based links which distinguish directions of the network traffic, the flow conservation requirements provide the linear model with a polynomial number of variables and constraints.

Similar approach to [30] is shown by Wu and Li in [45]. The authors employ conceptual flows and formulate linear programs with both edge and node capacity constraints for maximizing data rate allocation in overlay systems. Node capacity constraints are already modeled by both uplink and downlink access connection limits separately.

Linear programs derived from information flows with flow conservation yield optimal data rate allocation on overlay links, which specifies how much flow is to be routed through each link. A new question which appears then, is how to determine the content of these flows. Having optimal flows assigned to overlay links, Sanders et al. [35], Jaggi et al. [19] or Wang et al. [43] address the problem of actual flow allocation by assigning network codes. Such techniques require capability of nodes to perform efficient encoding and decoding operations, and even though there have been already proposed some polynomial time algorithms that design network codes to allocate actual flows, the use of re-encoding incurs a higher processing complexity at the nodes, which is less desirable from a practical

standpoint. Corresponding to a fundamental theorem of Edmonds [14], in case all nodes other than the source are destinations, the cut bound can be achieved by routing, therefore we do not need considering network coding in order to achieve the highest possible throughput when the overlay multicast system consists of a source and a set of end-hosts that can only replicate and forward received streams. However, the question of actual flows assignment still requires being addressed.

Therefore, referring to packing spanning or Steiner trees problems [4, 20], our previous works formulated fractional spanning trees problems dedicated to routing maximization in pure overlay multicast systems with node capacity constraints [25, 26]. These formulations incorporate a set of predefined routing trees. Considering such approaches with a given subset of trees that can be used, maximum possible flows might not reach volume as same as the volume that can be achieved globally, however achievable flow assignment is consistent without requirements of using additional techniques other than routing and flow allocation. Eventually, fractional spanning tree based approach is still a reasonable method that provides relative trade-off between computational time and result quality simultaneously.

3.1 Survivability aspects

Since state-of-the-art on network survivability are mainly focused on modeling and optimization of network-aware systems with unicast or anycast flows (e.g. [8, 33, 34, 42]) we are rather interested in pure overlays and multicast communication. Nevertheless, most of previous research on survivable overlay multicasting concentrates on the issues related to either special coding or providing many disjoint parents for each receiver. In addition, the path diversity is usually reduced to an assumption that the set of ancestors of a node in each tree should be as disjoint as possible [6, 31, 32].

The authors of [15] propose a proactive tree recovery mechanism. The objective is to make the overlay multicast resilient to peer failures. Simulations are applied to show that the proactive approach can recover from node failures much faster than reactive methods.

Eventually, a summary review is provided by Yang et al. [48] who present a survey of overlay multicast resilient approaches and classify them based on overlay topology design and error-resilient control approaches. However, overlay multicast techniques either do not allow optimization or consider only service continuity without aspects of throughput maximization.

In [40] Walkowiak presents a novel approach to survivability of overlay multicasting which aims at providing a completely survivable multicast routing by employing at least two multicast trees that carry the same copies of the content. The optimization objective is twofold, streaming cost and system throughput. Walkowiak formulates linear-based model and tackles the question of how additional survivability constraints guaranteeing failure-disjoint trees influence the routing cost and throughput of the peer-to-peer based multicasting system. In the paper, survivability definition concerns failures of the following network elements: overlay link, upstream node and ISP link. In the case of the overlay link failure, a pair of peers is disconnected. If there was a transfer on this link, some downstream nodes are affected by the failure. The second failure, uploading node failure, impacts all successors of the failed peer in the tree. Therefore, it is substantial to provide protection against failures of nodes that have some children. Leaf node failure affects only this one node. Eventually, overlay multicasting is usually used in the Internet, which consists of many ISP operators. Each peer is connected to a particular ISP. A failure of cross ISP link means, that all overlay links between peers of one ISP and peers of the second ISP are not available.

In [41] Walkowiak and Przewozniczek extend [40] and formulate survivable P2P multicasting with the streaming cost objective and propose algorithm for this novel problem. However, the survivability concerns protection methods implemented by redundancy and no other technique than linear approach for survivable multicast overlay with throughput maximization is given.

Next, in our previous work [24] we focus on slightly different survivability interpretation, namely we limit possible losses instead of guaranteeing redundant flows. The advantage of such a concept comprises providing higher data rates in the network, but every failure implies some data losses. In this work, two various linear approaches to overlay multicast flow maximization with survivability aspects are compared. The first is based on a generic data rate allocation, whereas following the aforementioned fact on fractional spanning tree based routing, the second uses predefined routing trees. Survivability constraints include disjointed trees where overlay links might be used in a given number of trees, virtual capacities on overlay links applied directly and relative link capacities that depend on overall achieved flow.

This paper extends [24] and four basic searches are designed and evaluated in functions of various versions of survivability in overlay multicast systems with a set of predefined routing trees.

4 Modeling: a linear approach

We consider an overlay system and the problem of information multicasting, when every user participating in the system obtains exactly the same amount of data from a single source.

Before discussing survivability constraints, we present an overlay multicast tree packing problem based on a set of predefined multicast spanning trees. In contrast to a generic version of the problem of an optimal data rate allocation in overlay multicast systems (e.g. [45]), here the linear formulation consists of a single set of variables representing data rates assigned to each overlay multicast tree.

We assume that multicast stream can be split into several separate substreams which might be illustrated as separate spanning trees rooted in the same vertex. Such a concept provides load in the network is balanced and users' resources are utilized in more effective way. Based on fractional Steiner tree packing problem proposed in [20] and overall discussion on packing spanning trees in [4], we employ a special case of trees packing by applying features of overlay networks in order to maximize multicast throughput in such systems. In contrary to Steiner tree packing and discussions covered in [4, 20], we address the problem of overlay multicasting comprising spanning trees (without Steiner nodes) with node capacity constraints instead of link capacity constraints.

The basic formulation includes a predefined set of trees $t \in \{1, 2, \ldots, T\}$ and a corresponding set of variables r_t, which describes flow allocated to tree t. The following version of a problem comprises trees represented by vector β which contains number of children of every node in each tree t. The problem is to find a maximal throughput assignment to predefined trees regarding available capacities of participants' access links. Every tree t represents a fractional multicast stream. Note that the vector β does not define an exact topology of every tree (actual arcs in the tree cannot be resolved) but such a tree representation is sufficient in order to formulate node capacity constraints and overlay multicast flows.

Fractional Spanning Tree Packing Problem (FST)

indices

$i, j = 1, 2, \ldots, V$ overlay nodes
$t \quad = 1, 2, \ldots, T$ predefined trees

constants (additional)

s source, root node ($s \in \{1, 2, \ldots, V\}$)
u_i upload capacity limit of node i
d_j download capacity limit of node j
β_{ti} number of i's children in tree t; 0 if i is a leaf

variable

r_t throughput assigned to tree t (e.g. streaming rate in kbps, pps)

objective

$$\max \; F = \sum_t r_t \tag{1}$$

constraints

$$\sum_t \beta_{ti} r_t \le u_i \quad \forall i \tag{2}$$

$$\sum_t r_t \le \operatorname{argmin}\left\{ \operatorname*{argmin}_{i \ne s}\{d_i\}, u_s \right\} \tag{3}$$

The objective function (1) maximizes summarized throughput of all fractional streams assigned to multicast spanning trees spread among V nodes with a single source of the content s. Next, node capacity constraints are introduced and each node i can neither contribute to the system exceeding its upload capacity nor download more than its download limit. Constraints (2) formulate upload limit based on available upload capacities u_i of every node in the system. Note that node i is a parent node in tree t, it uploads content of size r_t exactly β_{ti} times. Therefore i's total upload given by $\sum_t \beta_{ti} r_t$ cannot exceed its upload limit u_i. By analogy to upload bound (2), a set of constraints (3) is introduced in order to guarantee download limits of nodes are not surpassed. Basically, the download limit expresses that $\sum_t r_t$ cannot be greater than d_i, for every i. However, taking into account the overlay system, where all overlay nodes are connected to all trees, the actual throughput $\sum_t r_t$ is restricted not to exceed the minimal download limit d_i among all overlay nodes ($\operatorname{argmin}_{i \ne s}\{d_i\}$). Since we consider the system where source node s only uploads the content, the download capacity limit of s is not taken into consideration in this flow allocation problem. Moreover, a source's upload limit affects the total throughput $\sum_t r_t$ to be less or equal than u_s, i.e. all flows originate from root node s and the root cannot produce more throughput than its available upload limit u_s. Finally, a minimum value among $\operatorname{argmin}_{i \ne s}\{d_i\}$ or u_s limits overall throughput packed into the multicast system. In such a way a number of constraints that corresponds to the overall flow in the system is decreased to exactly one.

Note that in the abovementioned formulation all variables r_t may take both, continuous or integer values. While continuity of variables is not directly applicable in a majority of real networks, the models can serve as continuous relaxation for actual models, often used in algorithm development and as approximations.

5 Survivability constraints

Survivability as well as robustness are quite relative terms describing overall capability of the system to be resilient against specific types of failures and there are a lot of metrics for either expressing or designate robustness (e.g. [12]). In this part, we focus on survivability issues dedicated to overlay multicast system. The models express the overlay multicast system with survivability that limits potential losses of achievable throughput in case a single overlay link failure occurs.

5.1 Disjointed fractional trees

We assume a survivable overlay-based multicast topology to be relatively resilient against a single virtual (overlay) link failure, i.e., even in a case of link failure some of the flow can be still received by balancing it among various multicast trees and providing relatively disjointed substreams. Since a given vector β does not define an exact topology of a tree, overlay link-oriented survivability formulation requires a set of parent-relation constants. Here we formulate π_{ti} to define a parent node of i in tree t. Having π_{ti} then, actual topology of a multicast tree is unambiguously determined.

Fractional Spanning Tree Packing Problem with Link Disjoint Constraints

constants (additional)

D limit of common overlay links
π_{ti} parent node of i in tree t
M_t maximum throughput of tree t

variables (additional)

z_t $= 1$, if tree t is selected to form multicast topology; 0, otherwise (binary, auxiliary variable)

constraints (additional)

$$\sum_{t:\pi_{tj}=i} z_t \le D \quad \forall i,\ \forall j : j \ne i, j \ne s \tag{4}$$

$$r_t \le M_t z_t \quad \forall t \tag{5}$$

The simplest version of survivability constraints refers to limitation of using the same overlay link in different trees. The left-hand side of (4) is a number of chosen trees in which arc (i, j) is used and it cannot be greater than given limit D. Note that condition (5) is obligatory to provide r_t to be consistent, i.e. in Eq. (5), throughput bound M_t connects between r_t and z_t. In case $z_t = 0$, tree t cannot be assigned with any positive traffic (which implies $r_t = 0$), otherwise r_t cannot be greater than the maximum possible throughput of tree t, M_t, where M_t is defined by:

$$M_t = \min\left\{\operatorname*{argmin}_{i\ne s} d_i, \operatorname*{argmin}_{i:\beta_{ti}\ne 0} \frac{u_i}{\beta_{ti}}\right\} \quad \forall t \tag{6}$$

The upper bound of throughput of tree t (M_t) depends on either the minimal download limit of any node different than source s (a tree cannot realize more throughput than $\operatorname{argmin}_{i\ne s} d_i$) or upload limit of parent node i divided into number of its children.

Since we consider substream topology is based on a directed graph where actual flows are directed from parent to children nodes, arcs may be assumed to be asymmetric, i.e. overlay arc (i, j) differs from an overlay connection (j, i). This assumption remains true for real systems in which separate logical arcs might be realized on completely different paths through the physical network. Thus, we can narrow survivability meaning with regard to limit the use of overlay edges and modify condition (4) in the following way:

$$\sum_{t:\pi_{tj}=i} z_t + \sum_{t:\pi_{tj}=i} z_t \le D \quad \forall i < V,\ \forall j > i \tag{7}$$

where a solution is guaranteed to avoid more than D trees with common link including arcs in both directions either (i, j) or (j, i).

Such a set of survivability constraints cannot be easily applied to a generic version of data rate allocation in overlay systems because the formulation does not consider actual multicast flow allocation and dividing the content into several separate substreams realized on multiple various spanning trees will require additional techniques if this formulation is employed. Therefore, disjointed fractional trees constraints modeling survivability aspects in overlay multicast systems aimed at flow maximization is a domain of FST formulation.

5.2 Survivable flow redistribution: direct approach

Constraints (4) and (7) are related to quantity of links (arcs) with assigned throughput and ensure the overlay multicast system to be relatively robust against single link failure, i.e. in case an arc (edge) fails, at most D trees are eliminated from carrying the throughput (in the global meaning, i.e. even if some nodes still receive all flows injected by the source, from the system point of view, such a multicast throughput is not provided for all users). However, it is not defined how much throughput was assigned to failed trees and it is possible to lose most of designated throughput even if a single arc fails. Therefore, we propose quantity-of-throughput based survivable system which is characterized by limited data rates that can be realized via single overlay arc. Let D^* to be maximum throughput that might be carried between two adjacent nodes, then survivability is expressed by the following equations:

$$\sum_{t:\pi_{tj}=i} r_t \le D^* \quad \forall i,\ \forall j \ne i, j \ne s \tag{8}$$

$$\sum_{t:\pi_{tj}=i} r_t + \sum_{t:\pi_{ti}=j} r_t \le D^* \quad \forall i < V,\ \forall j > i \tag{9}$$

where (8) constraints the total throughput in all trees on arc from i to j not to exceed D^*, whereas (9) ensure D^*-survivability considering both directions of virtual link $i - j$ simultaneously.

In such a way, a single link failure may lead to traffic losses of volume D^* (e.g. kbps, pps). As long as a value

of D^* is given administratively and it is not related to total throughput in the system, in extreme cases the breakdown of arc traversed by throughput of D^* might cause significant loss of carried traffic. To tackle this potential drawback, we propose a concept of relative survivability.

5.3 Survivable flow redistribution: relative approach

To model a scenario with relative survivability and robustness against virtual link failures in overlay multicast systems, an additional survivability factor Ω is introduced. It describes a relation between the total throughput achieved in the system and the throughput that can be realized by a single arc or edge:

$$\sum_{t:\pi_{tj}=i} r_t \leq \Omega \sum_t r_t \quad \forall i,\ \forall j \neq i,\ j \neq s \tag{10}$$

$$\sum_{t:\pi_{tj}=i} r_t + \sum_{t:\pi_{ti}=j} r_t \leq \Omega \sum_t r_t \quad \forall i < V,\ \forall j > i \tag{11}$$

where Ω is selected between 0 and 1. The idea of relative survivability guarantees only Ω fraction of total data rate might be lost in case of a single link failure. Note that the constraints do not require using of auxiliary binary variables z as in disjointed fractional trees formulation.

Consequently, survivability constants might be link-dependent and defined by D_{ij}, D^*_{ij} or Ω_{ij}. In case some features of overlay arcs are rather certain or known in advance, the specialized constants lead to describe the system in more precise way with applying, e.g. different importance of arcs. Moreover, these constants, if set to 0, could define overlay bottlenecks or connections that cannot be realized in overlay system.

6 Heuristic and metaheuristic algorithms

In general, the performance of linear programming techniques may not often reach acceptable computational time while the size of the problem instance increases. In order to solve the maximum flow survivable overlay multicast with predefined routing trees with some result quality in reasonable time, we propose heuristic and metaheuristic algorithms. We begin with the version of the problem which comprises arc and edges survivability constraints defined in a direct way.

6.1 Random search

Random Search (RS) listed as Algorithm 1, is mostly randomized algorithm. It initializes a set of available trees T^* with given T spanning trees and a global variable d_{min} represents minimum available download limit among all receivers (line 3). Next, every overlay arc (i, j) is assigned

Algorithm 1 Random Search

```
1:  procedure RandomSearch(V, T, u, d, s, β, π, D*)
2:      T* ← T                                          ▷ initialization
3:      d_min ← argmin(d)
4:      for all i ≤ V − 1 do
5:          for all j > i do
6:              x_ij ← D*
7:              x_ji ← D*
8:          end for
9:      end for
10:     for all t ∈ T* do
11:         r_t^max ← MaximumFlow(t, x, u, d_min, β, π)
12:         r_t ← 0
13:     end for
14:     while T* ≠ ∅ do                                ▷ random assignment
15:         t ← SelectRandomTree(T*)
16:         r_t* ← AssignRandomFlow()
17:         r_t ← r_t + r_t*
18:         for all i do                                ▷ system state update
19:             u_i ← u_i − β_ti r_t*
20:             x_{π_ti≠s i} ← x_{π_ti i} − r_t*
21:         end for
22:         d_min ← d_min − r_t*
23:         for all t ∈ T* do
24:             r_t^max ← MaximumFlow(t, x, u, d_min, β, π)
25:             if r_t^max = 0 then
26:                 T* ← T* − t
27:             end if
28:         end for
29:     end while
30: end procedure
```

by its maximum obtainable flow. Regarding a survivability value D^* per each arc, auxiliary variables x_{ij} are initialized with D^* (lines 4–9). Last part of initialization sets actual flow variables r_t to 0, and r_t^{max} as maximum possible flow that can be realized via each tree $t \in T^*$ separately. Procedure `MaximumFlow` takes into account a topology of the tree, available upload and download limits of every overlay node and actual flow realized on every arc. Main loop of the RS algorithm (lines 14–29) performs random flow assignments to randomly selected routing trees. Until the set of available trees T^* is not empty, RS chooses t in a random way (line 15) and determines a flow value r_t^* to be assigned to tree t (line 16). Next, r_t is increased by r_t^*. Note that r_t^* is not greater than r_t^{max}. When new throughput has been assigned to an overlay multicast, the system requires update. Two `for` loops in lines 18–28 update state of the overlay system as follows: upload limits of each node i are decreased by already assigned flow r_t^* times number of i's children β_{ti} (e.g. if i is a leaf and β_{ti} equals to 0, its u_i remains unchanged), actual arcs flows are updated as well and having π_{ti} as a parent node of i, $x_{\pi_{ti}\neq s i}$ represents flows on arc to i from its parent in tree t. In line 22, the download limit is updated. Since all nodes other than

source s receive the same flow in every tree, only a single variable representing minimum allowable download is maintained.

Next, remaining flows r_t^{max} are amended and finally the available routing trees set T^* is revised in order to comprise only trees with r_t^{max} greater than 0. Note that r_t^{max} gets 0 in case either x_{ji} reaches 0 and t uses arc (j,i) or available upload limit u_i equals to 0 and node i is not a leaf in tree t or d_{min} is 0. In the latter case, there are no more trees with positive r_t^{max}.

6.1.1 *Maximum flow procedure*

Procedure `MaximumFlow` (Algorithm 2) computes remaining flow that can be realized via multicast tree t. Basically, such a flow depends on available upload and download capacities of overlay nodes. The first loop of the procedure (lines 3–7) specifies a minimum value of upload u_{min} among all parent nodes in t. Note that, u_{min} is a subject to u_i of node i and its children number β_{ti}. The second loop in lines 8–12 compares actual arc utilization given by variables x. If an arc can carry flow x related to survivability constraints less than actual u_{min}, u_{min} updates to x. `MaximumFlow` requires at most $2V-2$ operations in the worst case, where $V-1$ nodes are parents. In order to improve computational effectiveness, input vector u might refer to nodes which are parents in t ($\beta_{ti} > 0$).

Algorithm 2 Maximum Flow

```
1:  procedure MAXIMUMFLOW(t, x, u, d_min, β, π)
2:      u_min ← argmin(u)
3:      for all i : β_ti > 0 do
4:          if u_min > u_i/β_ti then
5:              u_min ← u_i/β_ti
6:          end if
7:      end for
8:      for all i ≠ s do
9:          if u_min > x_{π_ti i} then
10:             u_min ← x_{π_ti i}
11:         end if
12:     end for
13:     return argmin(u_min, d_min)
14: end procedure
```

6.2 Simulated annealing

In this part, we present a different generic probabilistic metaheuristic. Simulated Annealing (SA) also uses random-based movements, however it extends Random Search by defining neighborhood and providing a probabilistic decision which might allow avoiding its local minimum trap. This technique, initially proposed by Kirkpatrick [21], derives its name from mimicking the process in metallurgy, where atoms in a metal create various structures when its heated and then slowly cooled. In the end the structure of the metal has an optimal energy configuration. Generally, while SA cannot guarantee finding the optimum solution, it can often find a very good solution, even in the presence of large scale instances. For the problem of multicast flow assignment based on predefined multicast trees SA algorithm is likely to outperform pure RS approach in case no survivability constraints are applied [25]. Following this conclusion we design SA dedicated to solving such a problem with these additional conditions.

A pseudocode of the SA is presented in listing of Algorithm 3. Except of system's topology parameters, additional variables τ, α and τ_{min} are essential in annealing-based method. Initial temperature τ is gradually decreased by so-called cooling factor $0 < \alpha < 1$ in each iteration. The algorithm stops working after reaching τ_{min} threshold.

In the beginning, simulated annealing sets its initial solution flow r_t and system state comprising remaining upload (u_i), download (d_{min}) limits and actual arc flows x_{ij} in a pure random way (line 2). This initial solution becomes best known solution and is saved in vector r^{best} (line 3). Starting from line 4, annealing process begins. Until τ is high enough, r^n, u^n, d^n and x^n represent flow, overlay nodes upload and download limits and remaining arc flows of neighbor solution, respectively. Next, set T^* in lines 9–14 is initialized with multicast trees that carry non-zero flow.

6.2.1 *Neighborhood*

In the next step, simulated annealing searches for a neighbor of the solution which is a new solution of the problem produced after altering the given state in the following way. Considering set T^*, a new set T_n^* consisting of randomly selected trees from T^* is created (line 15). Therefore T_n^* includes trees, which flows are to be modified. In line 17, r_t^* gets random value of thorughput to be removed from tree t and next, to line 23 all system parameters of the neighbor solution are updated. Lines 25–27 actualize maximum remaining flow of every tree t among all predefined trees. After removing random flows from multicast trees, T^* requires updating which is performed in lines 28–33. In the next loop (34–49), new flows are assigned to the neighbor solution. Basically, it is performed as same as Random Search works, however consecutive trees selected from set T^* are subject to having a maximum value of remaining throughput r_t^{max} (line 35).

Algorithm 3 Simulated Annealing

1: **procedure** SA($V, T, u, d, s, \beta, \pi, D^*, \tau, \alpha, \tau_{min}$)
2: $r, x, u, d \leftarrow$ RandomSearch($V, T, u, d, s, \beta, \pi, D^*$)
3: $r^{best} \leftarrow r$
4: **while** $\tau > \tau_{min}$ **do** ▷ annealing process
5: $u^n \leftarrow u$
6: $d^n \leftarrow d$
7: $x^n \leftarrow x$
8: $r^n \leftarrow r$
9: $T^* \leftarrow \emptyset$
10: **for all** $t \in T$ **do**
11: **if** $r_t^n > 0$ **then**
12: $T^* \leftarrow T^* \cup t$
13: **end if**
14: **end for**
15: $T_n^* \leftarrow$ SelectRandomTrees(T^*) ▷ neighborhood
16: **for all** $t \in T_n^*$ **do**
17: $r_t^* \leftarrow$ SelectRandomFlow()
18: $r_t^n \leftarrow r_t^n - r_t^*$
19: **for all** i **do**
20: $u_i^n \leftarrow u_i^n + \beta_{ti} r_t^*$
21: $x^n_{\pi_{ti} \neq s\, i} \leftarrow x^n_{\pi_{ti} i} + r_t^*$
22: **end for**
23: $d^n_{min} \leftarrow d^n_{min} + r_t^*$
24: **end for**
25: **for all** $t \in T$ **do**
26: $r_t^{max} \leftarrow$ MaximumFlow($t, x^n, u^n, d^n_{min}, \beta, \pi$)
27: **end for**
28: $T^* \leftarrow \emptyset$
29: **for all** $t \in T$ **do**
30: **if** $r_t^n > 0$ **then**
31: $T^* \leftarrow T^* \cup t$
32: **end if**
33: **end for**
34: **while** $T^* \neq \emptyset$ **do** ▷ neighbor assignment
35: $t \leftarrow$ MaxRemainingThroughputTree(T^*)
36: $r_t^* \leftarrow$ AssignRandomFlow()
37: $r_t^n \leftarrow r_t^n + r_t^*$
38: **for all** i **do** ▷ system state update
39: $u_i^n \leftarrow u_i^n - \beta_{ti} r_t^*$
40: $x^n_{\pi_{ti} \neq s\, i} \leftarrow x^n_{\pi_{ti} i} - r_t^*$
41: **end for**
42: $d^n_{min} \leftarrow d^n_{min} - r_t^*$
43: **for all** $t \in T^*$ **do**
44: $r_t^{max} \leftarrow$ MaximumFlow($t, x, u, d_{min}, \beta, \pi$)
45: **if** $r_t^{max} = 0$ **then**
46: $T^* \leftarrow T^* - t$
47: **end if**
48: **end for**
49: **end while**
50: $\Delta E \leftarrow \sum_t r_t^n - \sum_t r_t$
51: **if** $e^{\frac{\Delta E}{\tau}} >$ random(0, 1) **then** ▷ acceptance
52: $r \leftarrow r^n$
53: $u \leftarrow u^n$
54: $d \leftarrow d^n$
55: $x \leftarrow x^n$
56: **if** $\sum_t r_t^{best} < \sum_t r_t$ **then**
57: $r^{best} \leftarrow r$
58: **end if**
59: **end if**
60: $\tau \leftarrow \alpha\tau$ ▷ cooling process
61: **end while**
62: **end procedure**

6.2.2 *Acceptance probability*

Having a neighbor solution r^n of r, the SA examines a difference between utility functions defined as $\Delta E = \sum_t r_t^n - \sum_t r_t$ (line 50) and accepts the transition from the r to r^n with probability given by $e^{\frac{\Delta E}{\tau}}$ (line 51), where τ is a global time-varying temperature. Note that, if the neighbor provides better multicast performance ($\sum_t r_t^n > \sum_t r_t$), $e^{\frac{\Delta E}{\tau}}$ is greater than 1, thus better solution is always accepted. On the other hand, since $e^{\frac{\Delta E}{\tau}}$ is invariably positive, there is still possibility of changing the solution with worse one. The probability of transition increases while $\sum_t r_t^n$ increases in comparison to $\sum_t r_t$ and temperature τ is high enough, i.e. for higher values of τ, the acceptance probability grows. Such a feature prevents the method from becoming stuck at algorithm's local minimum.

6.3 Remaining flow selection strategy

Remaining Flow Selection Strategy (RFSS) heuristic constructively packs flows on trees which are selected in accordance to their actual and remaining throughput and number of parent nodes as well. This approach does not rely on any probabilistic decision and returns always the same solution for every instance of the problem. Algorithm 4 illustrates a pseudocode of the RFSS. An additional parameter which is required by RFSS is δ representing percentage proportion of maximum available flow that is assigned to selected routing trees.

The algorithm is initialized in the same fashion as RS, i.e. a set of trees T^* contains all predefined trees, d_{min} represents minimum available download capacity limit among all receivers, variables x correspond to available flow per each overlay link, r_t^{max} is computed regarding Algorithm 2. Consecutive flow assignment is performed in loop 14–33. In contrary to random-based tree selection, here, tree t is chosen in correspondence to some logic comprising remaining throughput that can be assigned (r_t^{max}), throughput that is already assigned (r_t) and a number of parents in the tree ($\sum_i (\beta_{ti} > 0)$). The strategy of tree selection (`SelectTree`) is described in details in a pseudocode of Algorithm 5. After selecting a suitable t in line 15, the RFSS computes r_t^* as an δ portion of total available r_t^{max}. Such a strategy prevents the system resources from being exhausted too early (if r_t^{max} is assigned directly) and provides reasonable number of steps while throughput is allocated. In order to avoid infinite loops, a parameter ϵ is defined as a minimum possible flow to be assigned. Allocating a new flow requires system update (lines 22–32), where T^* contains only routing trees with non-zero r_t^{max}.

Algorithm 4 Remaining Flow Selection Strategy

```
1: procedure RFSS(V, T, u, d, s, β, π, D*, δ)
2:     T* ← T                                          ▷ initialization
3:     d_min ← argmin(d)
4:     for all i ≤ V − 1 do
5:         for all j > i do
6:             x_ij ← D*
7:             x_ji ← D*
8:         end for
9:     end for
10:    for all t ∈ T* do
11:        r_t^max ← MaximumFlow(t, x, u, d_min, β, π)
12:        r_t ← 0
13:    end for
14:    while T* ≠ ∅ do                                 ▷ flow assignment
15:        t ← SelectTree(T*, r, r^max, β)
16:        if r_t^max < ϵ then
17:            r_t* ← r_t^max
18:        else
19:            r_t* ← δ r_t^max
20:        end if
21:        r_t ← r_t + r_t*
22:        for all i do                                ▷ system state update
23:            u_i ← u_i − β_ti r_t*
24:            x_{π_ti≠s} i ← x_{π_ti} i − r_t*
25:        end for
26:        d_min ← d_min − r_t*
27:        for all t ∈ T* do
28:            r_t^max ← MaximumFlow(t, x, u, d_min, β, π)
29:            if r_t^max = 0 then
30:                T* ← T* − t
31:            end if
32:        end for
33:    end while
34: end procedure
```

Algorithm 5 Select Tree

```
1: procedure SELECTTREE(T*, r, r^max, β)
2:     r^m ← argmax(r^max)
3:     T^m ← 0
4:     for all t ∈ T* do
5:         if r_t^max = r^m then
6:             T^m ← T^m ∪ t
7:         end if
8:     end for
9:     t_x ← t ∈ T^m : argmin(r_t / Σ_i (β_ti > 0))
       return t_x
10: end procedure
```

The most important part of the Remaining Flow Selection Strategy is `SelectTree` procedure (Algorithm 5). It starts from setting its internal variable r^m equals to maximum r^{max} of all trees in T^* (line 2). Next, it examines every tree in order to find all multicast trees from T^* with $r_t^{max} = r^m$ (lines 4–8). At this stage, T^m includes all trees with equal and maximum remaining throughput. Finally, among all trees in T^m, `SelectTree` chooses the one with minimum value of actual throughput divided into number of parents in the tree (line 3). Such a selection technique provides the procedure with picking trees either with small value of r_t (and balancing the flow among many separate trees since it prefers trees with small throughput realized on them) or with large number of nodes that have predecessors (in order to limit possible losses of nodes upload capacities). Since r_t^{max} is derived from x and node capacity limits, `SelectTree` procedure does not require such input parameters other than the presented in the pseudocode.

6.4 Hybrid search

Hybrid Search (HS) consists of two basic steps. In the beginning HS uses Remaining Flow Selection Strategy in order to provide initial feasible solution. Next, it reallocates the flow in a random way. Listing of Algorithm 6 shows a pseudocode of the Hybrid Search. In line 2, a start solution is determined by using RFSS with δ parameter. In the second part of the procedure (starting from line 3), the HS rearranges this initial throughput allocation to allow its convergence to maximum flows. Note that, except of overlay network topology, δ required by RFSS, an additional parameter ι must be guaranteed as an input to the HS. This parameter is an integral value and represents a number of trials where the rearrangement part of the algorithm may not improve its current solution. In such a way, ι determines a condition of finishing of the Hybrid Search. Till line 13, the algorithm prepares auxiliary variables that represent initial solution to be rearranged. Then a set T^* contains all trees with non-zero traffic being allocated. In line 14 `SelectRandomTrees` creates a new set T_n^* by choosing randomly some trees from T^*. This step is equivalent to neighborhood selection in simulated annealing, where flows on all trees in a new set T_n^* are supposed to be modified. Next, in loop 16–25 random flows are subtracted from trees of T_n^* and an internal variable y tracks total reduced throughput (line 24). Since then, some upload as well as download limits and available overlay links capacities increase, therefore a new set of possible flow allocation appears. In lines 30–34 a set T^* is prepared. T^* comprises trees of which maximum available data rate r_t^{max} is greater than total throughput subtracted in the previous step (y). In case $T^* \neq \emptyset$, the previous solution might be improved by at least $r_t^{max} - y$. Next, until $T^* \neq \emptyset$, the flows are reallocated in a random way (lines 35–50). Finally, based on a simple result comparison in line 51, a new solution is saved as the best solution found and if the random rearrangement process has not improved the current solution, ι decreases in line 57. The HS stops working after ι reaches 0.

Algorithm 6 Hybrid Search

```
1: procedure HybridSearch(V, T, u, d, s, β, π, D*, δ, ι)
2:     r, x, u, d ← RFSS(V, T, u, d, s, β, π, D*, δ)
3:     while ι > 0 do
4:         u^n ← u
5:         d^n ← d
6:         x^n ← x
7:         r^n ← r
8:         T* ← ∅
9:         for all t ∈ T do
10:            if r_t^n > 0 then
11:                T* ← T* ∪ t
12:            end if
13:        end for
14:        T_n* ← SelectRandomTrees(T*)
15:        y ← 0
16:        for all t ∈ T_n* do                    ▷ flow substraction
17:            r_t* ← SelectRandomFlow()
18:            r_t^n ← r_t^n − r_t*
19:            for all i do
20:                u_i^n ← u_i^n + β_ti r_t*
21:                x^n_{π_ti≠s i} ← x^n_{π_ti i} + r_t*
22:            end for
23:            d^n_min ← d^n_min + r_t*
24:            y ← y + r_t*
25:        end for
26:        for all t ∈ T do
27:            r_t^max ← MaximumFlow(t, x^n, u^n, d^n_min, β, π)
28:        end for
29:        T* ← ∅
30:        for all t ∈ T do
31:            if r_t^max > y then
32:                T* ← T* ∪ t
33:            end if
34:        end for
35:        while T* ≠ ∅ do                        ▷ flow rearrangement
36:            t ← SelectRandomTree(T*)
37:            r_t* ← AssignRandomFlow()
38:            r_t^n ← r_t^n + r_t*
39:            for all i do                       ▷ system state update
40:                u_i^n ← u_i^n − β_ti r_t*
41:                x^n_{π_ti≠s i} ← x^n_{π_ti i} − r_t*
42:            end for
43:            d^n_min ← d^n_min − r_t*
44:            for all t ∈ T* do
45:                r_t^max ← MaximumFlow(t, x, u, d_min, β, π)
46:                if r_t^max = 0 then
47:                    T* ← T* − t
48:                end if
49:            end for
50:        end while
51:        if Σ_t r_t^n > Σ_t r_t then
52:            r ← r^n
53:            u ← u^n
54:            d ← d^n
55:            x ← x^n
56:        else
57:            ι ← ι − 1
58:        end if
59:    end while
60: end procedure
```

6.5 Edge-oriented survivability

All aforementioned algorithms with their listings present versions for arc-oriented survivability. While considering edge-defined survivability, actual flows in one direction affect usability of the edge in the opposite direction, compare constraints (4) and (7), (8) and (9), (10) and (11). In such a way, every heuristic or metaheuristic requires a simple modification. For instance in random search, line 20 is extended with an additional condition and instead of

20: $x_{\pi_{ti\neq s}i} = x_{\pi_{ti}i} - r_t^*$

there is reciprocity as follows:

20: $x_{\pi_{ti\neq s}i} = x_{\pi_{ti}i} - r_t^*$
21: $x_{i\neq s\pi_{ti}} = x_{i\pi_{ti}} - r_t^*$

where r_t^* represents traffic currently added to tree t and $x_{\pi_{ti}i}$ is an arc to i from its parent π_{ti}, thus $x_{i\pi_{ti}}$ is an arc from i to π_{ti}. Note that, if any positive flow is allocated to arc (i, π_{ti}) in tree t it may not affect its opposite direction in practice, in case other trees do not contain arc (π_{ti}, i).

Similarly, SA, RFSS and HS are modified.

6.6 Relative throughput survivability

Previous subsections cover algorithms dedicated to maximization of multicast flow based on predefined spanning trees in overlay networks with survivability defined as direct overlay link capacity constraints. In this part, we propose and discuss a general approach to the problem with relative throughput survivability. In such a case, arc-oriented or edge-oriented limits are strongly related to total maximum flow obtainable in the network. Algorithm 7 describes an universal procedure of multicast flow assignment with relative throughput survivability (Multicast Flow Binary Search—MFBS). Since overlay links usage depends on achievable flow, we apply a binary search (half-interval search) to find a maximum feasible throughput that can be allocated in the overlay system with respect to Ω-based constraints. Algorithm MFBS is initialized with the following values: r^{best} tracks the best found objective value, r^n refers to current value of the objective, y represents actual directed arc or edge capacity limit and y changes with respect to y^{max} and y^{min}. The two latter values bound a feasible set of ys to be examined. In the beginning, y gets a minimum value of source node upload capacity u_s or minimum download limit among all receivers d_{min} in line 4 and y^{max} is equal to y. Since then, y represents survivability throughput limit D^*. Regarding current value of D^* (y) the survivable multicast problem is solved in loop 7–21 by using one of the previously defined algorithm. In line 8, Random Search, Simulated Annealing, Remaining Flow Selection Strategy or Hybrid Search determine maximum feasible flow r^n subject to survivability throughput limit $D^* = y$. Next, unless

Algorithm 7 Multicast Flow Binary Search

```
1:  procedure MFBS(V, T, u, d, s, β, π, Ω)
2:      r^best ← 0
3:      r^n ← 0
4:      y ← argmin(u_s, d_min)
5:      y^max ← y
6:      y^min ← 0
7:      while y^max − y^min > ϵ do
8:          r^n ← Algorithm(...)
9:          if r^n = 0 then return
10:         else
11:             if y^min / r^n ≤ Ω then
12:                 y^min ← y
13:                 if r^best < r^n then
14:                     r^best ← r^n
15:                 end if
16:             else
17:                 y^max ← y
18:             end if
19:         end if
20:         y ← y^min + (y^max − y^min)/2
21:     end while
22: end procedure
```

positive r^n is found the algorithm stops working. This assumption is derived from the fact that multicast flow is said to be generally nondecreasing function of D^*. Note that, an initial value of y creates the problem equivalent to survivability unconstrained (limits of D^* are sufficiently high to be relaxed). In case achievable actual r^n is greater than 0, an essential condition of the MFBS technique is tested in line 11. Namely, a relation of y to r^n must satisfy Ω constraint. If so, the space of possible D^*s is lower bounded by $y^{min} = y$ in line 12, and if r^{best} is updated accordingly to its relation to r^n (line 14). While positive r^n does not meet the condition in line 11, y's upper bound is set to $y^{max} = y$ (line 17). In each iteration y gets its value corresponding to half-interval steps and the formula in line 20. The MFBS terminates when the difference $y^{max} - y^{min}$ falls below a given ϵ (which is usually a very small value).

7 Results

This section provides comparison of the performance of fractional spanning trees based model and heuristic algorithms for maximum flow survivable overlay multicast with survivability constraints. The computational experiments were carried out on an Intel Core 2 Duo CPU with 2.13 GHz clock and 4GB RAM, with x64 Windows 7 Professional. All algorithms were implemented in C++ under MS VS2008. Optimization tools used for linear-based optimization include *IBM ILOG CPLEX Optimization Studio Academic Research Edition 12.3* and its C++ interface (see [18] for the former version).

We examine an instance of overlay multicast systems comprising 100 nodes connected to the network by ADSLs. Source node can upload 1536 kbps, whereas available upload limits of access links of other nodes are proportionally distributed among values 512, 1024 and 1536 kbps. For the sake of simplicity, we assume download capacities $d_i = d_{min} = 4096$ kbps are the same for all nodes in the system, thus only upload limits and survivability constraints may imply bottlenecks in the system. A set of predefined trees consists of $T = 5000$ separate multicast spanning trees generated randomly.

7.1 Survivable flow redistribution (D^*)

First experiments comprise the problem with arc oriented survivability limit D^* and to compare the efficiency of heuristic and metaheuristic searches, optimal benchmarks of FST formulation were obtained with CPLEX. Table 1 shows results for maximum flow survivable overlay multicast with predefined routing trees with arc and edge defined survivabilites, respectively. Column D^* is expressed in kbps and represents capacity limits of virtual links between every pair of overlay nodes and "FST" presents optimal benchmarks. Columns "result" refer to average achievable objective value in kbps obtained in 100 independent runs of the searches and each column "sec." shows average computational time in seconds elapsed while one run of every algorithm. Every "result" is accompanied by its relative gap to FST-based optimum (columns "gap"). Moreover, because of random nature of algorithms we list standard deviations (in kbps) for HS, SA and RS in order to observe their actual efficiency.

In case D^* limits flows on overlay arcs, relaxing an upper bound of D^* impacts on faster convergence of the FST for which obtaining optimal results requires about 10–40 seconds if D^* is greater than 20 kbps and linear techniques might provide results faster than even random draws of solution. Although, the FST linear formulation has been shown as a good trade-off between computational time and result quality in comparison to data rate allocation linear model for multicast throughput maximization with survivability constraints for instances of smaller sizes with 30–40 nodes and 1000–5000 trees (see [24]), for instances with 100 nodes and 5000 trees it is unable to find any feasible solution within 1 hour computation if D^* is less than 6 kbps. Therefore, if D^* is strongly limited, every other algorithm is more efficient than the linear-based FST approach.

The only algorithm without random-based decisions is Remaining Flow Selection Strategy (RFSS). Table 1 refers RFSS results with parameter δ set to 0.1. This implies every tree selected in accordance to its remaining capacity, actual flow and a number of parent nodes is assigned by 10 % of maximum flow that can be allocated. The value of δ is tuned among 0.01–1.0 and with $\delta = 0.1$ the RFSS yields the best

Table 1 Optimization results for D^* based direct survivability

	D^*	FST		Hybrid Search		RFSS ($\delta = 0.1$)		Simulated Annealing		Random Search	
		result	sec.	result (std, gap %)	sec.	result (gap)	sec.	result (std, gap)	sec.	result (std, gap)	sec.
arc constraints	2	x	3600	93.3 (0.3, x)	31	86.5 (x)	17	87.7 (0.4, x)	79	50.5 (1.4, x)	10
	10	524.3	152	474.7 (2.0, 9.5)	32	431.8 (17.6)	16	445.4 (0.5, 15.0)	112	256.8 (9.2, 51.0)	12
	20	697.0	99	639.0 (1.6, 8.3)	32	535.5 (22.3)	16	569.0 (3.7, 18.4)	24	419.6 (15.5, 39.8)	13
	30	735.8	12	684.5 (1.1, 7.0)	33	540.0 (26.6)	14	569.0 (18.1, 22.7)	15	419.6 (20.6, 43.0)	12
	40	760.4	34	700.4 (2.6, 7.9)	30	541.3 (28.8)	15	600.0 (18.2, 21.1)	15	419.6 (13.9, 44.8)	12
	50	776.4	9	712.2 (2.2, 8.3)	31	541.3 (30.3)	14	627.8 (28.5, 19.1)	14	419.6 (35.8, 46.0)	11
	100	793.7	11	727.5 (0.5, 8.4)	30	541.3 (31.8)	14	719.45 (10.6, 9.4)	15	419.6 (37.9, 47.1)	12
edge constraints	2	80.7	429	70.0 (0.01, 12.3)	33	68.3 (15.4)	15	65.6 (0.4, 18.6)	78	41.9 (1.4, 48.0)	15
	10	403.5	467	348.4 (0.02, 13.7)	33	341.6 (15.3)	14	327.7 (2.3, 18.8)	108	202.6 (19.7, 49.8)	14
	20	649.2	112	591.9 (3.1, 8.8)	34	502.9 (22.5)	15	496.4 (2.2, 23.5)	24	362.3 (14.4, 44.2)	14
	30	707.9	264	648.7 (0.01, 8.4)	33	502.9 (29.0)	15	496.4 (5.8, 29.9)	17	365.0 (11.6, 48.4)	15
	40	739.5	62	678.6 (2.2, 8.2)	34	502.9 (32.0)	15	517.5 (21.7, 30.0)	16	365.0 (23.1, 50.6)	16
	50	761.1	91	699.9 (2.6, 8.0)	34	502.9 (33.9)	14	613.5 (14.5, 19.4)	16	379.9 (32.9, 50.1)	14
	100	793.3	497	723.1 (2.5, 8.9)	32	502.9 (36.6)	14	718.8 (15.8, 9.4)	16	410.2 (30.8, 48.3)	14

results. Considering Hybrid Search which is initialized with RFSS solution, δ also equals to 0.1. For the HS, an additional parameter ι needs setting and the experimentations were carried out with $\iota = 1000$, i.e. the algorithm reallocated its current flow and finished after 1000 tries without improvement of the result. More challenging is to determine all input parameters of simulated annealing algorithm with regard to efficient searching in reasonable time or search space. First of all, if an initial temperature τ is too low, it might constraints transistions to the neighbour states (solutions), hence local minimum traps might appear. On the other hand, if τ is to high, the convergence of the algorithm may get pure random-like behavior. Next, since we use geometric cooling schedule, where τ decreases in geometric progression by multiplying the previous value by a fixed cooling factor α and the SA runs until reaching τ_{min}, the parameters have an influence on operational time of the metaheuristic. Wide experimentations allowed us to choose the following values of the SA inputs: $\tau = 100.0$, $\alpha = 0.65$ and $\tau_{min} = 0.001$. Such parameters guarantee a relative trade-off between required computational time and the quality of solutions yielded by the SA, where computational time grows geometrically in relation to the value of α.

Left-hand side of Fig. 2 illustrates results of different approaches in function of $2 \leq D^* \leq 100$ kbps. Hybrid Search improves its RFSS initial solution by applying a kind of random local search procedure which reallocates flows of the solution. By doing so, the HS outperforms RFSS in the meaning of result quality and provides multicast flows about 100–170 kbps better, decreasing the gap to optimality even by 20 %. However, it multiplies twice required computational time in relation to the RFSS. Regarding operational time, random search yields its results in the fastest way but such results of random flow assignment fall below 40–50 % optimal values. Simulated annealing requires almost 2 minutes in order to return solutions for survivability constraints D^* up to 10 kbps. It indicates there are relatively a lot of shifts from current solutions to the neighbor ones which cause more operations. If D^* is greater than 10 kbps, SA provides results within time comparable to RFSS. Multicast routing achievable by SA exceeds both random search and the RFSS and for $D^* < 18$ kbps it differs form HS solution in about 30 kbps in the whole system. This gap increases for $18 < D^* < 62$ kbps and reaches even 100 kbps but if D^* is more relaxed and gets 70 kbps and more, the results of SA approach to HS and reaches about 10 % gap to optimums, whereas results of HS guarantee the gap at level of 8.4 % by their average. Standard deviation of the Hybrid Search solutions in 100 experiments falls even below 0.1 % of achieved flow, thus it seems to be an algorithm with quite stable results. Simulated annealing results variability exceeds 10–20 kbps for greater values of D^* which implies about 3–4 % difference in its maximized flow. Eventually, 20–30 and 10–40 kbps of standard deviation yielding

approximately 3–8 % of the multicast survivable flows are provided by RFSS and RS, respectively.

In addition, we present RFSS with $\delta = 1$, i.e. maximum possible flow is packed on the selected multicast tree, and RS with fully utilized trees capacities while every next tree is drawing to the final solution ("RS max flow"). However, these curves mark only a kind of lower multicast survivable throughput thresholds because realizing maximum achievable throughput on each consecutive tree leads to fast capacity resources exhaustion. Moreover, in such a case even a cost-derived heuristic RFSS yields approximately random-like results, hence, even though they return solutions within 10–11 seconds by average, it is not efficient to allocate maximum possible throughput on trees, since it leads to sudden elimination of both available upload capacities of parent nodes and available throughput of overlay links and the final multicast flow is reduced.

Slightly different time relation appears if edge-defined survivability constraints are applied (lower part of Table 1 and right-hand side diagram in Fig. 2). Since flows on overlay arcs also affect their usability in opposite direction, from a practical point of view the problem is more constrained. It has an influence on increased computational time required by a linear approach modeled to the FST formulation. On the one hand, the FST returns optimal results for D^* less than 6 kbps (in contrast to arc-oriented constraints), but on the other hand it needs about 10–40 times more seconds for providing optimums for greater values of D^*. For these cases, other algorithms overtake the FST in the meaning of computational time, which is commonly similar to the results for the problem with arc-oriented survivability definition (computational time insignificantly grows only for RS).

Taking into consideration the effectiveness of survivable multicast flows yielded by different algorithms, the overall relation of gap to optimality might be said to remain almost similar to the arc-oriented version. It only slightly grows for HS (1 %) and equals to 8–14 %, RFSS (3–4 %) is 15–37 %, SA (about 4 %) returns 9–30 % results and random search solution deterioration reaches to 44–51 % in comparison to 40–51 % for arc-constrained survivability. In addition, in contrary to previous relations, where simulated annealing exceeds RFSS results through the entire domain, here, starting from $D^* = 34$ solutions obtained with the SA overcome the constructive search of Remaining Flow Selection Strategy with δ parameter set o 0.1.

Since edge-oriented survivability constraints also limits the size of the solution space which impacts on possible movements while solving the problem instances, the results of HS gain more stability for $D^* \leq 10$ kbps but the gap to optimum increases, e.g. from 9.5 to 13.7 % (for $D^* = 10$). The diversity of fully random based RS and SA is quite similar to previous results.

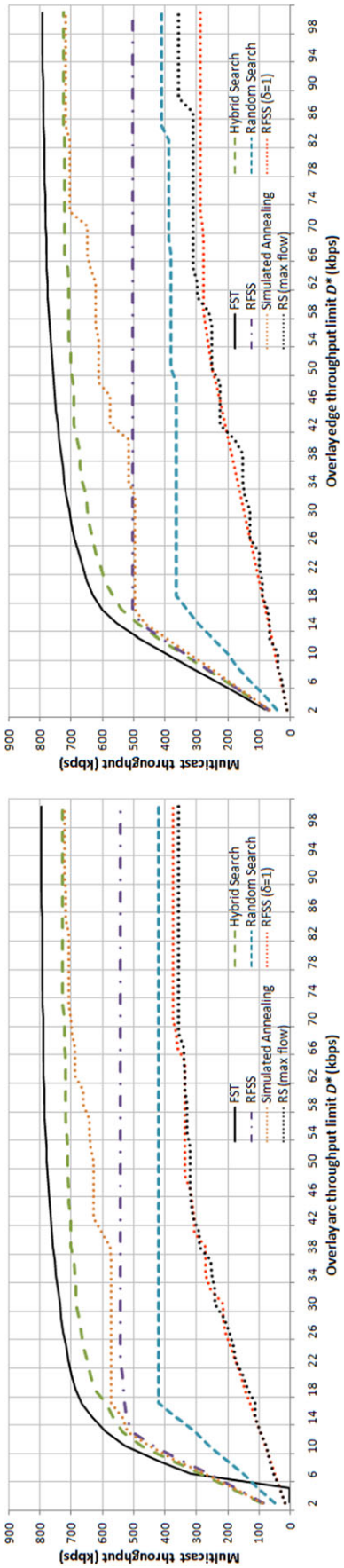


Fig. 2 Direct survivability: arc- (*left*) and edge-oriented (*right*) scenarios

To conclude, the average effectiveness of presented algorithms, RFSS yields 15–37 % gap to optimum, the Hybrid Search improves the RFSS initialized results to about 7–14 % gap to optimum, random algorithm returns only 40–51 % efficiency of optimally maximized flows and the SA which is an improvement over pure random search provides 9–30 % gap, depending on the survivability limits in the networks. Although linear-based approach modeled as the FST is still efficient technique, in the meaning of computational time, if survivability value D^* is greater than 30 kbps and applied to one-way direction of overlay links, on the other hand, heuristics gain more applicability if the survivability constraints limit both direction of overlay links simultaneously.

7.2 Relative survivability Ω

In this part, we illustrate and discuss results of linear FST approach versus heuristic and metaheuristic searches applied to Multicast Flow Binary Search technique.

Intuitively, if relative survivability with parameter Ω is taken into account in the maximum multicast flow problem, it becomes more and more difficult. First of all, separate tree flow variables r_t are relatively dependent on the total value of $\sum_t r_t$, and in case of D^* survivability where overlay links are virtually capacitated it is possible to guarantee a solution with non-zero mutlicast throughput routed via only a single multicast tree, for relative version of survivability it is, however, beyond the bounds of possibility if Ω is less than 1. From the practical point of view, there is no reasonable goal to set $\Omega = 1$ becasue in case 100 % of total alloacted flow might be realized on every overlay link in the network, it is directly equivalent to survivability unconstrained version of the problem. Thereby, we need more trees than one to allocate positive flows and satisfy relative survivability constraints.

In addition, CPLEX solver and the FST formulation encountered an out-of-memory error while dealing with $V = 100$ and $T = 5000$ instance for the problem with relative Ω survivability, thus solving large size instances is often ineffective, computational time required for performing optimization becomes unacceptable or impractical and finding any feasible solution leads to be often impossible. In such a case "gap" relation refers to results provided by the best search method—the Hybrid Search.

In the beginning, it is worth pointing out the growth in computational times required for performing optimization. In contrary to survivability constraints applied directly by limiting overlay links capacities with D^*, the problem with relative survivability condition needs 10–15 times longer period in order to provide a feasible solution. It is derived from the fact that the MFBS technique designed for this version of the overlay multicast flow problem multiple times calls every of the selected algorithm. Table 2 and Fig. 3 show optimization results for the problem with relative form of survivability constraints, where ϵ is set to 0.1, i.e. if the range of values D^* consecutively applied to the MFBS falls below 0.1 kbps, algorithms stop working. By having such an ϵ and upload capacity limit of the source u_s equal to 1536 kbps, the MFBS runs every algorithm at most 15 times. Eventually, in order to provide consistent comparison, all parameters applied to algorithms are the same as in the previous case.

Results show that, the HS not only returns the best results but also is the stablest algorithm (among algorithms with random-based operations) in the meaning of standard deviation, even though it consumes much more time than the others. If arc-oriented survivability is taken into consideration (left-hand side of Fig. 3), simulated annealing reaches to only 2 % gap in relation to HS, while RFSS provide 21–26 % gap and pure random technique yields its results with even 50 % gap. An interesting observation can be noticed for $\Omega \geq 16$ % from which average allocated flows obtained by RS strategy with 100 % of tree remaining flow assignment exceed average results of RS with random flow assignment. However, even such method of random flow assignment is far from being a real competitor for other algorithms.

The quality of HS's results increases for edge-based survivability conditions (right-hand side of Fig. 3). Here, SA efficiency drops to about 12 % gap and 20–30 % gap is obained by RFSS. Random search guarantees almost the same effectiveness with 50 % gap to the solutions yielded by hybrid search.

8 Conclusion

This paper presents the problem of survivable multicasting in overlay networks which are aimed at flow maximization where flows are realized on predefined multicast routing trees. The survivability constraints concern on failures of single virtual (overlay) connections. The main goal of the survivable multicast routing is to provide limited losses in case of link failure while overall throughput of the system is maximized. Three various interpretations of ensuring survivability in the overlay multicast network are modeled and discussed. First, disjointed spanning tree constraints limits the utilization of trees with common overlay arcs or edges. Second, overlay links are virtually capacitated in order to balance flows and limit their usage. And finally, relative survivability factor is introduced. The factor bounds overlay arcs or edges traffic in accordance to already achieved multicast flows. The problem and all survivability versions are formulated into linear programs which derived from fractional Steiner trees and employ fractional spanning tree

Table 2 Optimization results for Ω based relative survivability

	Ω %	FST		Hybrid Search		RFSS ($\delta = 0.1$)		Simulated Annealing		Random Search	
		result	sec.	result (std)	sec.	result (gap %)	sec.	result (std, gap)	sec.	result (std, gap)	sec.
arc constr.	2	x	3600	0 (0)	457	0 (0)	250	0 (0, 0)	675	0 (0, 0)	180
	10	x	3600	727.9 (0.6)	469	539.7 (25.9)	190	715.6 (9.1, 1.7)	276	361.8 (8.5, 50.3)	166
	20	x	3600	729.8 (2.6)	490	539.7 (26.1)	194	715.6 (24.8, 2.0)	273	361.8 (2.5, 50.4)	173
	30	x	3600	729.8 (2.1)	492	566.5 (22.4)	192	715.6 (34.5, 2.0)	268	393.2 (8.6, 46.1)	176
	40	x	3600	729.8 (2.6)	485	575.2 (21.2)	199	715.6 (7.1, 2.0)	270	406.2 (4.6, 44.3)	171
edge constr.	2	x	3600	0 (0)	514	0 (0)	284	0 (0, 0)	622	0 (0, 0)	182
	10	x	3600	716.4 (1.6)	509	498.4 (30.4)	199	637.5 (8.6, 11.0)	223	360.2 (1.6, 49.7)	188
	20	x	3600	726.1 (1.6)	501	498.4 (31.4)	189	637.5 (4.5, 12.2)	235	360.2 (2.2, 50.4)	176
	30	x	3600	726.1 (1.3)	511	558.8 (23.0)	190	637.5 (3.6, 12.2)	232	411.2 (6.9, 43.4)	171
	40	x	3600	726.1 (1.0)	519	558.8 (23.0)	194	637.5 (1.9, 12.2)	228	431.8 (3.4, 40.5)	177

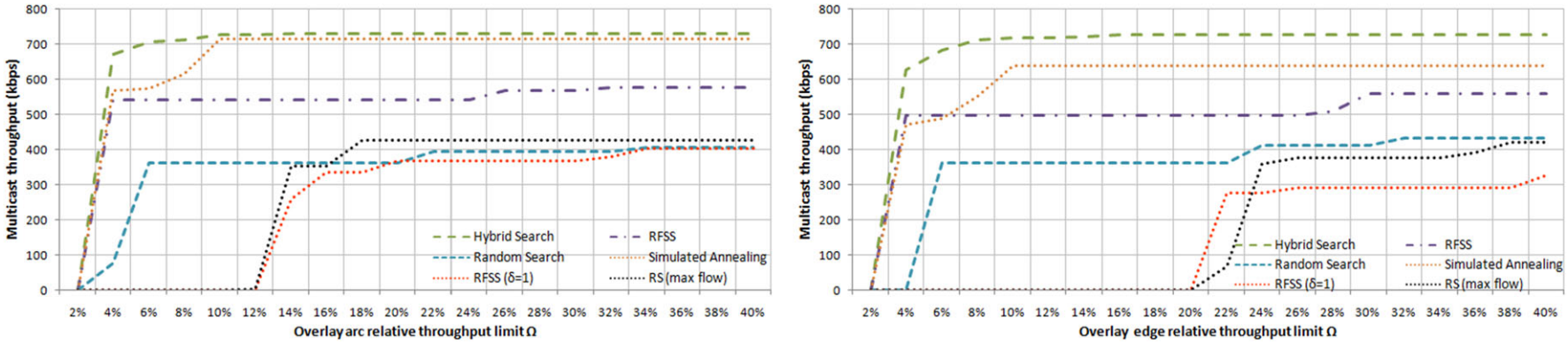


Fig. 3 Relative survivability: arc- (*left*) and edge-oriented (*right*) scenarios

packing (FST) with additional linear constraints. All programs consist of polynomial number of variables and constraints as well. Since the dimension of the FST problem depends on the number of predefined trees and the network size, thus using linear-based approach for larger instances may become even impossible due to limited either time or computing resources. In order to solve the survivable multicast routing in overlay systems with virtually capacitated overlay links, we propose various techniques including heuristic and metaheuristic searches. To examine random behavior of overlays, random search (RS) is designed. Results provided with the RS may mark a kind of lower bounds. An improvement over random search is applied to simulated annealing (SA). This well discussed approach, by simulating physical process observed in metallurgy, tries to reallocate some flows in order to enhance pure random results. Next, a constructive search Remaining Flow Selection Strategy is proposed. The algorithm considers remaining capacities of trees and made its flow assignment decision in accordance to utility comprising actual flows and number of parent nodes in different trees. This heuristic does not use random decisions. The last method, hybrid search (HS) improves the RFSS by applying additional local search which is performed in a random way. The performance of all algorithms is evaluated in relation to optimal results and in case no optimal results are obtained, to the results of the HS.

Experimentations show that, the most efficient algorithm, in the meaning of result quality is hybrid search, however it requires more computational time than the others. The relative gap to optimal of the HS results does not exceed 7–14 % which explicitly outperforms SA (9–30 %), RFSS (15–37 %) and RS (39–51 %).

Concerning relative survivability that constraints the multicast network much stronger because certain arc flow affects its requirements on the total throughput in the system. For such a survivability problem, the linear approach is unable to provide any feasible results. In order to use all designed algorithms a general Multicast Flow Binary Search procedure (MFBS) is proposed. It solves the problem with survivability constraints applied to overlay links and then examines relative survivability conditions. In case the conditions are not satisfied, the MFBS changes actual direct survivability constraints accordingly to half-interval step as it is performed with binary search. For the overlay multicast flow problem with relative survivability constraints, random search technique cannot provide very efficient results and falls to even 50 % of effectiveness of hybrid search. Slightly better approach is RFSS, the results of which reach 20–30 % gap to HS and while HS takes advantage of the RFSS for initializing its starting solution, it improves the the results by about 100–200 kbps in the presented overlay system. A relative competitor of HS is SA, which yields 2 % or 11–12 % gap to results of the HS (depending on arc or edge applied survivability limit) but it requires approximately twice less operational time.

Finally, results indicate that, it is worth applying local searches that are even based on random behavior in order to improve result quality and decrease the gap to optimal upper bounds. Moreover, well-designed heuristics or metaheuristics are much more applicable in large scale practical systems.

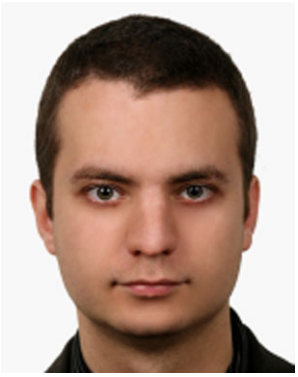
Acknowledgements Michał Kucharzak is supported by the European Social Fund according to the Operational Programme Human Capital *National Cohesion Strategy*.

The work was supported by the Polish National Science Centre (NCN), under the grant N N519 650440.

References

1. Ahlswede, R., Cai, N., Li, S. Y. R., & Yeung, R. W. (2000). Network information flow. *IEEE Transactions on Information Theory*, *46*(4), 1204–1216.
2. Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Upper Saddle River: Prentice-Hall.
3. Akbari, B., Rabiee, H. R., & Ghanbari, M. (2008). An optimal discrete rate allocation for overlay video multicasting. *Computer Communications*, *31*(3), 551–562. doi:10.1016/j.comcom.2007.08.025.
4. Barahona, F. (1995). Packing spanning trees. *Mathematics of Operations Research*, *20*, 104–115. doi:10.1287/moor.20.1.104. URL http://dl.acm.org/citation.cfm?id=215008.215020.
5. Benslimane, A. (Ed.) (2007). *Multimedia multicast on the Internet*. ISTE.
6. Birrer, S., & Bustamante, F. E. (2005). Resilient peer-to-peer multicast without the cost. In *Proc. of MMCN*.
7. Cayley, A. (1889). A theorem on trees. *Quarterly Journal of Mathematics*, *23*, 376–378.
8. Cetinkaya, E., Broyles, D., Dandekar, A., Srinivasan, S., & Sterbenz, J. (2011). Modelling communication network challenges for future internet resilience, survivability, and disruption tolerance: a simulation-based approach. *Telecommunication Systems*. doi:10.1007/s11235-011-9575-4.
9. Hua Chu, Y., Rao, S. G., & Zhang, H. (2000). A case for end system multicast. In *Proceedings of ACM sigmetrics* (pp. 1–12).
10. Cui, Y., Xue, Y., & Nahrstedt, K. (2003). Optimal resource allocation in overlay multicast. In *Proc. of 11th international conference on network protocols, ICNP 2003*.
11. Deering, S. E., & Cheriton, D. R. (1990). Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, *8*, 85–110.
12. Dekker, A. H., & Colbert, B. D. (2004). Network robustness and graph topology. In *Proceedings of the 27th Australasian conference on computer science, ACSC '04* (pp. 359–368). Darlinghurst: Australian Computer Society, Inc.

13. Diot, C., Levine, B. N., Lyles, B., Kassem, H., & Balensiefen, D. (2000). Deployment issues for the IP multicast service and architecture. *IEEE Network*, *14*(1), 78–88.
14. Edmonds, J. (1965). Minimum partition of a matroid into independent sets. *Research of the NBS*, *69*, 67–72.
15. Fei, Z., & Yang, M. (2007). A proactive tree recovery mechanism for resilient overlay multicast. *IEEE/ACM Transactions on Networking*. *15*(1), 173–186. doi:10.1109/TNET.2006.890086.
16. Garg, N., Khandekar, R., Kunal, K., & Pandit, V. (2003). Bandwidth maximization in multicasting. In *Proceedings of European symposium on algorithms* (pp. 242–253).
17. Grötschel, M., Martin, A., & Weismantel, R. (1996). Packing steiner trees: polyhedral investigations. *Mathematical Programming*, *72*, 101–123. doi:10.1007/BF02592085. URL http://dl.acm.org/citation.cfm?id=228038.228039.
18. IBM ILOG CPLEX 12.1: user's manual for CPLEX (2009).
19. Jaggi, S., Sanders, P., Chou, P. A., Effros, M., Egner, S., Jain, K., & Tolhuizen, L. M. G. M. (1973–1982). *Polynomial time algorithms for multicast network code construction*, *51*(6), 1973–1982. doi:10.1109/TIT.2005.847712.
20. Jain, K., Mahdian, M., & Salavatipour, M. R. (2003). Packing steiner trees. In *SODA '03: proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 266–274). Philadelphia: SIAM.
21. Kirkpatrick, S. (1983). Optimization by simulated annealing. *Numerische Mathematik*, 4598.
22. Koetter, R., & Medard, M. (2003). An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, *11*(5), 782–795. doi:10.1109/TNET.2003.818197.
23. Kucharzak, M., & Walkowiak, K. (2010). Optimization of flows in level-constrained multiple trees for p2p multicast system. In *The second international conference on advances in P2P systems, AP2PS 2010*, Florence, Italy.
24. Kucharzak, M., & Walkowiak, K. (2011). Fractional spanning tree packing problem with survivability constraints for throughput maximization in overlay multicast networks. In *Proc. 3rd international ultra modern telecommunications and control systems and workshops (ICUMT) congress* (pp. 1–7).
25. Kucharzak, M., & Walkowiak, K. (2011). An improved annealing algorithm for throughput maximization in static overlay-based multicast systems. In *Proceedings of the 6th international conference on hybrid artificial intelligent systems, HAIS'11, part I* (pp. 364–371). Berlin: Springer. URL http://dl.acm.org/citation.cfm?id=2021442.2021490.
26. Kucharzak, M., & Walkowiak, K. (2011). On modelling of fair throughput allocation in overlay multicast networks. In *NEW2AN* (pp. 529–540).
27. Lao, L., Hong Cui, J., & Gerla, M. (2005). Multicast service overlay design. In *Proc. of second international symposium on wireless communication systems, ISWCS'05*, Philadelphia, PA, USA.
28. Leuf, B. (2002). *Peer to peer: collaboration and sharing over the Internet*. Boston: Addison-Wesley Longman.
29. Li, S. Y. R., Yeung, R. W., & Cai, N. (2003). Linear network coding. *IEEE Transactions on Information Theory*, *49*(2), 371–381. doi:10.1109/TIT.2002.807285.
30. Li, Z., Li, B., & Lau, L. C. (2006). On achieving maximum multicast throughput in undirected networks. *IEEE Transactions on Information Theory*, *52*(6), 2467–2485. doi:10.1109/TIT.2006.874515.
31. Liang, J., & Nahrstedt, K. (2006). Dagstream: locality aware and failure resilient peer-to-peer streaming. In *SPIE multimedia computing and networking (MMCN 2006)*.
32. Padmanabhan, V. N., Wang, H. J., & Chou, P. A. (2003). Resilient peer-to-peer streaming. In *Proc. 11th IEEE international network protocols conference* (pp. 16–27).
33. Rak, J. (2010). k-Penalty: a Novel approach to find k-disjoint paths with differentiated path costs. *IEEE Communications Letters*, *14*(4), 354–356. doi:10.1109/LCOMM.2010.04.091597.
34. Rak, J., & Walkowiak, K. Reliable anycast and unicast routing: protection against attacks. Telecommunication Systems. doi:10.1007/s11235-011-9583-4.
35. Sanders, P., Egner, S., & Tolhuizen, L. (2003). Polynomial time algorithms for network information flow. In *Proceedings of the fifteenth annual ACM symposium on parallel algorithms and architectures, SPAA '03* (pp. 286–294). New York: ACM.
36. Sentinelli, A., Marfia, G., Gerla, M., Kleinrock, L., & Tewari, S. (2007). Will iptv ride the peer-to-peer stream? Peer-to-peer multimedia streaming. *IEEE Communications Magazine*, *45*(6), 86–92.
37. Shi, S., & Turner, J. S. (2002). Multicast routing and bandwidth dimensioning in overlay networks. *IEEE Journal on Selected Areas in Communications*, *20*, 1444–1455.
38. Small, T., Li, B., Member, S., & Liang, B. (2007). Outreach: Peer-to-peer topology construction towards minimized server bandwidth costs. *IEEE Journal on Selected Areas in Communications*, *25*, 35–45.
39. Walkowiak, K. (2009). Network design problem for P2P multicasting. In *International network optimization conference, INOC 2009*.
40. Walkowiak, K. (2009). Survivability of p2p multicasting. In *Proc. 7th international workshop on design of reliable communication networks, DRCN 2009* (pp. 92–99).
41. Walkowiak, K., & Przewoniczek, M. (2011). Modeling and optimization of survivable p2p multicasting. *Computer Communications*, *34*, 1410–1424. doi:10.1016/j.comcom.2010.12.011.
42. Walkowiak, K., & Rak, J. (2011). Simultaneous optimization of unicast and anycast flows and replica location in survivable optical networks. *Telecommunication Systems*. doi:10.1007/s11235-011-9611-4.
43. Wang, M., Li, Z., & Li, B. (2005). A high-throughput overlay multicast infrastructure with network coding. *Lecture Notes in Computer Science*, *3552*, 37–53.
44. Wu, C., & Li, B. (2005). Optimal peer selection for minimum-delay peer-to-peer streaming with rateless codes. In *P2PMMS'05: proceedings of the ACM workshop on advances in peer-to-peer multimedia streaming* (pp. 69–78). New York: ACM. doi:10.1145/1099384.1099394.
45. Wu, C., & Li, B. (2007). Optimal rate allocation in overlay content distribution. In *Networking* (pp. 678–690).
46. Wu, C., & Li, B. (2008). On meeting p2p streaming bandwidth demand with limited supplies. In *Proc. of the fifteenth annual SPIE/ACM international conference on multimedia computing and networking*.
47. Wu, G., & Chiueh, T. (2005). *Peer to peer file download and streaming* (rpe report, tr-185).
48. Yang, H., Hu, R., Chen, J., & Chen, X. (2008). A review of resilient approaches to peer-to-peer overlay multicast for media streaming. In *Proc. 4th international conference on wireless communications, networking and mobile computing, WiCOM '08* (pp. 1–4). doi:10.1109/WiCom.2008.802.
49. Zhu, Y., & Li, B. (2008). Overlay networks with linear capacity constraints. *IEEE Transactions on Parallel and Distributed Systems*, *19*(2), 159–173. doi:10.1007/s11235-013-9823-x.



Michał Kucharzak received his M.Sc. Eng. in teleinformatics from Wroclaw University of Technology in 2008. He is currently a Ph.D. candidate at Wroclaw University of Technology and holds a position of Senior Radio Research Engineer at Wroclaw Research Center EIT+, Poland. In recent years, he has been a member of reviewer committees for many international journals, program and technical committees for various conferences as well. His current research interests are primarily in the areas of network modeling and network optimization with special regard to overlays, simulations, design of efficient algorithms and wireless system protocols.

Krzysztof Walkowiak was born in 1973. He received the Ph.D. degree and the D.Sc. (habilitation) degree in computer science from the Wroclaw University of Technology, Poland, in 2000 and 2008, respectively. Currently, he is an Associate Professor at the Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology. His research interest is mainly focused on optimization of network distributed systems like P2P systems, multicasting systems, Grid systems; optimization of connection-oriented networks (MPLS, DWDM); network survivability; application of soft-optimization techniques for design of computer networks. Prof. Walkowiak was involved in many research projects related to optimization of computer networks. Moreover, he was consulting projects for the large companies including TP SA, PZU, PKO BP, Energia Pro, Ernst and Young, Skanska. Prof. Walkowiak published more than 150 scientific papers. He serves as a reviewer for many international journals including: Computer Communications, Future Generation Computer Systems, Computational Optimization and Applications, International Journal of Applied Mathematics and Computer Science, Expert Systems, Pattern Analysis and Applications, International Journal of Computer Mathematics. He is/was actively involved in many international conferences. Prof. Walkowiak is a member of IEEE and ComSoc.