Construction of the Bounded Application-layer Multicast Tree in the Overlay Network Model by the Integer Linear Programming[*]

Petr Jurčík
Department of Control Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
jurcikp@control.felk.cvut.cz

Zdeněk Hanzálek
Department of Control Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
hanzalek@fel.cvut.cz

Abstract

The geographically distributed system can be interconnected via a overlay multicast network. In this overlay network the multicast data are routed and replicated on the application layer along a multicast tree. This paper presents the techniques of the network reduction and the multicast tree construction. The multicast tree in the form of shortest path tree (SPT) can be build up upon the linear programming formulation. To control the load of each host, the additional constraints on the maximal number of directly outgoing connections and integer variables are added and subsequently form the degree-bounded shortest path tree problem (db-SPT). This theoretically based problem is formulated in integer linear programming framework.

Keywords: Application-layer multicast, overlay network, shortest path tree problem, degree-bounded shortest path tree problem, integer linear programming.

1. Introduction

Many of today's industrial applications are geographically distributed and therefore their efficient interconnection is realized via Internet. These applications often require time optimal multicast communications.

This paper presents construction of a multicast tree where the number of outgoing connections from each node is limited. The objective is to minimize a delay measured from the moment when the multicast message is sent from the source node to the moment when the message is delivered to the last node in a multicast group. Such problem is NP-hard as shown by reduction from Traveling Salesman Problem (TSP) in Section 2.

The algorithms maintaining the multicast tree [10] are usually far from the optimum [9]. Especially when performing join and leave operations of separate nodes, the value of their optimality criterion degrades dramatically. That is why this paper focuses on optimal construction of the tree using integer linear programming, which is not intended to be used during dynamic behavior of the network (joining/leaving nodes, changing network parameters) but it is used by so called "reshaping procedure", which builds new optimal multicast tree when executed.

Multicasting [3] is an efficient mechanism for data delivery in one-to-many data transfer applications. It eliminates a redundant data packet replication in the network. The unicast and multicast packets are formally the same, but the processing of these data packets by the routers is different. The unicast packet is forwarded by the router, according to the routing table, to only one appropriate next hop in the path from the source to the destination. The multicast packet is forwarded by the router to the one or more next hops therefore the packet must be replicated by this router. The routers need to know the optimal paths for transmission and replication of the data packets to all subscribers belonging to an appropriate multicast group. These paths form a multicast tree. The advantage of multicast communication is that the sender sends data only to the multicast group and the network itself makes a replication of the packets in the optimal places. This way of data transfer reduces a global load of a transmission network and a load of the source node.

Unlike native multicast where data packets are replicated on the network layer at routers inside a network, in application-layer multicast [8] data packets are replicated on the application layer at end hosts. The host is a router performing the multicast algorithm on the application layer. All hosts belong to a multicast group and form an overlay network over a physical network. The multicast data are routed and replicated only on this overlay network. On the physical network the routers handle the multicast data packets in the same way as the unicast data packets i.e. they only route the data packets along the optimal way. Since the application-layer

[*] This work has been supported by the Ministry of Industry and Trade of Czech Republic under project 61-03001.

multicast must sometimes send the identical multicast packets over the same link in the physical network and it has higher overhead while processed on the application layer, it is less efficient than native multicast. On the other hand, the application-layer multicast algorithm is a software which can be controlled, configured and modified by the user. Contrariwise the native multicast algorithm running on the network layer is controlled only by the network administrator and it can not be modified except some special cases. The access on the application layer opens a room for the experiments and it is much easier than the access on the network layer.

In order to achieve efficient application-layer multicast communication (i.e. without many hosts directly connected to the root in a multicast tree), the load of each host in the overlay multicast network should be controlled. The load of the host corresponds to the amount of a dataflow that the host must process (route). Each host can limit the number of outgoing connections (out-degree bound) in the multicast tree thereby it indirectly limits its load.

The ball-and-string model of the network topology and the linear programming formulation of the SPT problem on this network model are presented in [12]. In [11] is shown the dual linear programming approaches to the SPT problem and relationships among them. The network layer multicast [3] is not widely adopted by most commercial ISPs, and thus large parts of Internet are still incapable of native multicast. Application layer multicast protocols [8], [13] do not change a network infrastructure and they implement multicast functionality exclusively at end hosts (i.e. the overlay network).

The paper is organized as follows. Section 2 presents the graph representation of the network, graph reduction procedure and the degree-bounded shortest path tree (db-SPT) problem formulation. Section 3 presents the adopted solution of the shortest path tree (SPT) problem by linear programming. The solution of db-SPT problem by Integer Liner Programming (ILP) is described in Section 4. Section 5 summarizes the experimental results and Section 6 concludes the work.

2. Problem statement

2.1. Graph representation of the network

The physical network can be represented by a *directed, strongly connected, weighted graph G(N,E)*, where N is a set of nodes with cardinality n (i.e. $|N|=n$), and E is a set of edges with cardinality m (i.e. $|E|=m$). The *directed edge* from node i to node j is denoted as $e_{i,j}$ and it is weighted with $a_{i,j}\in \boldsymbol{R}^{+}$. Therefore each edge $e_{i,j}\in E$, representing communication link from router i to router j, is labeled by a real non-negative value $a_{i,j}$ called *weight*, which can represent a communication delay in a physical network. The *path* $P_{i,j}$ is a connected sequence of nodes and edges without duplication from node i to node j. The *weight of the path* $w(P_{i,j})$ is given as a sum of weights of traversed edges in the path $P_{i,j}$.

A *tree* T is a connected, acyclic sub-graph of graph G: $T\subseteq G(N,E)$. A tree on v nodes contains exactly *(v-1)* edges. Every two nodes of a tree are connected by a unique path [1]. A *spanning tree* of the graph *G(N,E)* is a subset of *(n-1)* edges that form a tree. A *directed out-tree*, rooted at node r (called the *root*), is a tree where the unique path from node r to every other node is a directed path [2].

The edges and nodes, in a graph representation, correspond to the communication links and routers in a physical network. A mapping of a multicast communication on Internet-like topology can be efficiently represented by the directed out-tree (the multicast tree) where the root corresponds to a *multicast source*.

2.2. Graph reduction

The physical network topology of one autonomous system can be obtained from a link-state database of underlying link-state internet routing protocol (e.g. OSPF [4]). Full topology of geographically wide network is hard to acquire. But when constructing an overlay multicast network (Figure 1 (b)), it is sufficient to measure the Round Trip Time (RTT) among several hosts in different autonomous systems.

There are two types of nodes in the graphic representation of a physical network denoted by graph G', i.e. the multicast nodes and the transport nodes.

The *multicast node* can be source or destination of the multicast communication and it performs the routing and replication of multicast data in an optimal way. The application-layer algorithms performing the multicast communication run only in these nodes. The multicast node is called a *host* in a network specification. The multicast nodes can be configured and controlled by the administrator of a multicast group i.e. by us.

The *transport nodes* are neither source nor destination of the multicast communication. These nodes make no distinction between multicast and non-multicast types of communication and they only route an incoming messages using standard unicast routing algorithms (like OSPF, RIP). The transport nodes can be configured and controlled only by the administrator of the network i.e. not by us. The transport nodes represent general routers in a network specification.

The multicast nodes form an *overlay network* over the physical communication network. This overlay multicast network can be represented by the graph G where the transport nodes are reduced and each edge from multicast node i to multicast node j reflects the shortest path from node i to node j over reduced transport nodes.

The modified Floyd's algorithm is used in our reduction procedure as illustrated in Figure 1. The original version of Floyd's algorithm investigates whether the shortest path from node i to node j goes via node k. Therefore the original algorithm deals with 3 nested loops, where the outermost loop varies index k

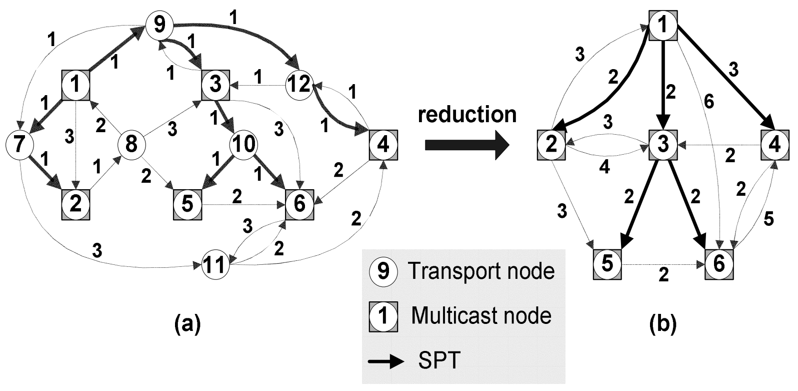

Figure 1: Transport nodes are labelled by the circle and the multicast nodes are additionally labelled by the dark square. The bold edges represent the SPT. (a) Graph G' is the graphic representation of the physical network. (b) The reduction procedure transforms the physical network to the overlay multicast network graphically represented by the graph G.

over all nodes in a graph. In our case we look for the shortest paths only over the transport nodes, therefore the index *k* varies over the transport nodes but not over the multicast nodes.

The reduction procedure is illustrated in Figure 1, where the physical network with 11 nodes is reduced to the overlay multicast network with 6 nodes. We are able to deal with large physical networks, since the graph reduction procedure is polynomial ($O(n^3)$).

In practical applications we assume to have about tens of hosts (e.g. 20 geographically distinct locations of a water distribution company) in the "part of Internet" with thousands of routers (transport nodes).

2.3. Shortest path tree problem

The *Shortest Path Tree* (SPT) of graph *G* is such spanning, directed out-tree which has the least weights of paths from the root to any other node in a graph (Figure 1(b)). Note that there can be more SPTs in a graph.

In our case of time-optimal multicast communication, the weight $a_{i,j}$ represents the communication delay, i.e. the time from the start of the transmission of the message from node *i* to the end of the reception of the message in node *j*.

Classical algorithms solving the SPT problem include Bellman-Ford algorithm [1] and Dijkstra's algorithm [2] with asymptotic complexity of $O(n.m)$ and $O(n^2)$, respectively. Slightly more general is Floyd's algorithm [2], solving the all pairs shortest path problem. It computes shortest path distances between all pairs of nodes in graph *G* in $O(n^3)$ time.

The SPT problem can be also formulated as an optimization problem in a linear programming (LP) framework, solvable in polynomial time. The advantage of using linear programming approach is its flexibility, i.e. when the problem statement is extended by new constraints in many cases we just simply add new variables and inequalities.

2.4. Degree-bounded shortest path tree problem

To use the SPT for Internet based application-layer multicast is not realistic, since the resulting SPT has many nodes directly connected to the root (due to underlying physical network maintained by OSPF protocol) and it does not consider different performance of different nodes. Moreover, when intensive communication is considered, such root centric tree with unbalanced load of separate nodes would likely change the timing parameters of underlying physical network. Since the load of host corresponds to the amount of processed dataflow, it is useful to limit the number of outgoing connections (i.e. *out-degree bound* in a graph terminology or *fanout* in [10]) in the multicast tree. Thereby the host indirectly limits and controls its load. Consequently the multicast tree must satisfy the requirements of the shortest path tree while satisfying the out-degree bounds of each host.

Therefore we extend the above SPT problem by the out-degree bounds as follows. The *degree-bounded Shortest Path Tree* (db-SPT) of graph *G* is such spanning, directed out-tree which satisfies the out-degree bounds of all nodes and minimizes the maximal weight of the path from the root to any other node in the graph (Figure 2). Constant *outDb(i),* the out-degree bound of node *i*, denotes the maximal number of outgoing edges from node *i*. Therefore *outDb(i)* ranges from *0* to $|E^+(i)|$, where $E^+(i)$ denotes a set of outgoing edges from node *i*.

Unfortunately the degree-bounded shortest path tree problem is NP-complete, as can be shown by the reduction from the Traveling Salesman Problem (TSP), which is known to be NP-complete. We show, how each instance of the TSP can be reduced to an instance of the db-SPT problem as follows: The instance of the TSP is given as a search of minimum weight Hamiltonian cycle (i.e. oriented closed path where each node is traversed exactly once) in complete graph *H*. We choose the node *r* to be the root. We create graph *G* as an extension of graph *H* by adding a copy of the root, called *r'*, such that $a_{i,r'} = a_{i,r}$ and $a_{r',i} = a_{r,i}$ for all $i \neq r$. The TSP on graph *H* can be solved while formulating the db-SPT problem on graph *G* where *outDb(i)=1* for all $i \neq r'$ and *outDb(r')=0*. Therefore the spanning tree found by the db-SPT problem is the minimum weight path from node *r* to node *r'*, since each node (excluding the node *r'*) has exactly one successor. This tree path corresponds to the minimum weight Hamiltonian cycle while considering the nodes *r* and *r'* as one node, i.e. this path gives the solution of the TSP. The path is the most restricted form of a tree and hence the other trees are generalization of above problem and are harder to solve.

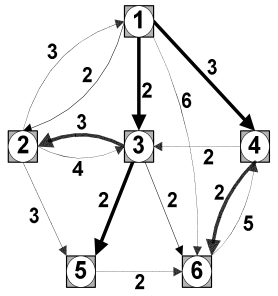

Figure 2 : Example of the degree-bounded shortest path tree (db-SPT) with out-degree bound (*outDb*) for all nodes equal to 2. The bold edges represent the db-SPT.

3. Solution of shortest path tree problem by linear programming

3.1. Objective criterion

In this section, the linear programming approach is used to solve the SPT problem, since this formulation is further extended to db-SPT problem in the next section. The objective of the SPT problem is to minimize the delays (the weights of paths) from the root to each of other nodes in the graph *G,* which indeed minimizes the maximal delay. In a multicast communication, the maximal delay (i.e. the worst-case delivery time of the multicast message) is given as time between an instant when the source sends out the message and an instant when the last host of the multicast group receives this message. Therefore the objective criterion in multicast communication is formalized as

$$\min\left(\max_{j\in\{N\text{-}\{r\}\}} \mathrm{w}(P_{rj})\right). \qquad (1)$$

Please notice that several paths in multicast tree can go over one edge, although in the multicast communication there is at most one message traversing the edge.

Remark: In contrast to the SPT, another optimization problem called *Minimum Spanning Tree* (MST) minimizes a sum of communication costs while considering the weight of edge to be the communication price. Therefore the problem of MST is to find such spanning tree T_{MST} with the minimum sum of weights of the edges formalized as

$$MST\text{:}\ \min\left(\sum_{\forall e_{i,j}\in T_{MST}} a_{i,j}\right). \qquad (2)$$

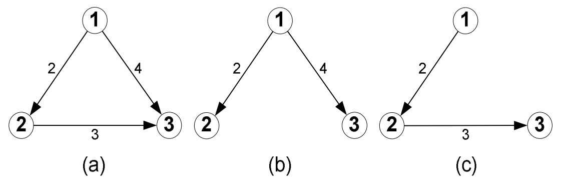

Figure 3: Illustration of a difference between the SPT and MST problem. (a) Graph G. (b) The spanning, directed out-tree of G rooted at node 1. The value of max{2,4}=4 is the minimal value of criterion (1). On the other hand, the value of (2+4)=6 is not the minimal value of criterion (2). Therefore this tree is the SPT but it is not the MST. (c) Another spanning, directed out-tree of G rooted at node 1. The value of max{2, 2+3}=5 is not the minimal value of criterion (1). On the other hand, the value of (2+3)=5 is the minimal value of criterion (2). Therefore this tree is not the SPT but it is the MST.

3.2. Linear programming formulation

Let $\mathrm{SPT_j}$ is the sub-tree of SPT rooted at node *j*. The value of the linear programming variable $x_{i,j}$ associated with the edge $e_{i,j}$ represents the number of nodes in $\mathrm{SPT_j}$. In other words variable $x_{i,j}$ represents the number of paths of SPT traversing edge $e_{i,j}$.

$$x_{i,j} \begin{cases} > 0 & \text{if } e_{i,j} \in \mathrm{SPT(G)} \\ = 0 & \text{if } e_{i,j} \notin \mathrm{SPT(G)} \end{cases}$$

In the multicast communication problem, the value of $x_{i,j}$ denotes the number of hosts served via the communication link from host *i* to host *j*. Therefore variable $x_{i,j}$ achieves only the integer values although there is no restriction on the values in the linear programming formulation. There are as many $x_{i,j}$ variables as there are edges in graph *G* (i.e. there are *m* variables in this LP formulation).

In this specific case the extreme of objective criterion (1) is equivalent to the minimal sum of weights of the paths from the root to each of other nodes as

$$SPT\text{:}\ \min\left(\sum_{j\in\{N\text{-}\{r\}\}} \mathrm{w}(P_{rj})\right). \qquad (3)$$

Remark: For example we consider the graph *G* from Figure 3(a). The value (2+4)=6 is the minimal value of the objective criterion (3) for the tree in Figure 3(b). Therefore this tree is the SPT. The value (2+(2+3))=7 is not the minimal value of criterion (3) for another tree in Figure 3(c). Therefore this tree is not the SPT.

Therefore the objective criterion can be formulated in its linear form by equation (3). Then the linear programming (LP) formulation of the shortest path tree problem is the following [2]:

$$\min \sum_{e_{i,j}\in E} a_{i,j} x_{i,j} \qquad (4)$$

subject to:

$$\sum_{\forall e_{i,j}\in E^{+}(i)} x_{i,j} + 1 = \sum_{\forall e_{j,i}\in E^{-}(i)} x_{j,i} \qquad \forall i \in \{N-r\} \qquad (5)$$

$$\sum_{\forall e_{i,j}\in E^{+}(r)} x_{r,j} = n-1 \qquad (6)$$

$$x_{i,j} \geq 0 \qquad \forall e_{i,j} \in E \qquad (7)$$

Figure 4 : LP formulation of the shortest path tree (SPT) problem.

The constrains (5), (6) ensure that the set of selected edges, given by variables $x_{i,j}$, is the spanning tree. The objective criterion (4) corresponds to the one given in (3) (we obtain the minimal sum of weight of the paths); therefore it ensures that this spanning tree is the SPT.

In the multicast communication problem, the source serves *n-1* hosts, as denoted by the constraint (6) where $E^{+}(r)$ corresponds to the set of the outgoing communication links from the source (i.e. the root of graph *G*). Consequently constraint (5) denotes that the number of hosts served via outgoing links from host *i* (excluding the source, i.e. $i \neq r$) plus 1 (the host *i* itself) is equal to the number of hosts served via incoming links to host *i*. $E^{+}(i)$ corresponds to the set of outgoing communication links from host *i* and $E^{-}(i)$ corresponds to the set of incoming communication links to host *i*.

4. Solution of degree-bounded shortest path tree problem by integer linear programming

As stated above, the number of communication connections can be restricted in the db-SPT problem. In addition to the LP formulation of SPT problem in Figure 4, new variables are added so that the linear combination of these variables could be restricted by *outDb(i)*, the constant giving the number of outgoing communication connections from node *i*. Therefore we add a binary variable $y_{i,j}$ determining whether the edge from node *i* to node *j* is in the db-SPT (multicast tree) or not. Considering variable $x_{i,j}$ to be defined as the one in the previous section, we can deduce the value of $y_{i,j}$ out of the value of $x_{i,j}$ as follows. If and only if variable $x_{i,j}$ is greater than *0* (the edge $e_{i,j}$ is a part of the tree) then the binary variable $y_{i,j}$ is equal to *1*, otherwise (the edge $e_{i,j}$ is not a part of the tree) variable $y_{i,j}$ is equal to *0*. Therefore

$$x_{i,j} = 0 \iff y_{i,j} = 0 \qquad \forall e_{i,j} \in E, \qquad (8)$$

$$x_{i,j} > 0 \iff y_{i,j} = 1 \qquad \forall e_{i,j} \in E. \qquad (9)$$

The values determined by these relations are graphically represented by square symbols in Figure 5. The maximum number of outgoing edges from each node is *n-1* (i.e. $x_{i,j} \leq n$-1).

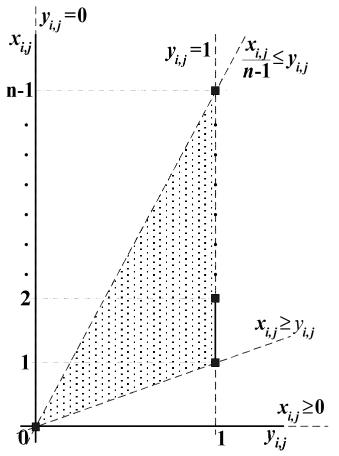

Figure 5: A mapping between variable $x_{i,j}$ and variable $y_{i,j}$.

Considering binary values of variable $y_{i,j}$, relations (8), (9) can be formulated in the form of two linear inequalities (10) and (11) illustrated in Figure 5.

$$x_{i,j} \geq y_{i,j} \qquad \forall e_{i,j} \in E \qquad (10)$$

$$\frac{x_{i,j}}{n-1} \leq y_{i,j} \qquad \forall e_{i,j} \in E \qquad (11)$$

Using variable $y_{i,j}$, we are able to formulate the following out-degree constraint for each node *i* as

$$\sum_{\forall e_{i,j}\in E^{+}(i)} y_{i,j} \leq outDb(i) \qquad \forall i \in N. \qquad (12)$$

The following constraint (13) ensures that the found tree is spanning tree.

$$\sum_{\forall e_{i,j}\in E} y_{i,j} = n-1 \qquad (13)$$

After addition of binary variable $y_{i,j}$ to the LP formulation of the SPT problem (Figure 4) the formulation has changed to the non-convex ILP and the extreme of objective criterion (1) is no more the same as the extreme of criterion (3). Therefore we can not further use the criterion (3) and that is why we add new variable $v_{i,j,k}$ for formulation of the criterion (1). A binary variable $v_{i,j,k}$ determines whether edge $e_{i,j}$ is on the shortest path $P_{r,k}$ from the root *r* to node *k*. The value of variable $v_{i,j,k}$ is equal to 1 if and only if the corresponding edge $e_{i,j}$ is on the path $P_{r,k}$, otherwise $v_{i,j,k}$ is equal 0. There are *(n-1)m* variables $v_{i,j,k}$.

For a given edge $e_{i,j}$, the sum of the values of variable $v_{i,j,k}$ determines the number of the shortest paths traversing this edge which corresponds to the meaning of variable $x_{i,j}$. Therefore

$$\sum_{k\in\{N-r\}} v_{i,j,k} = x_{i,j} \qquad \forall e_{i,j} \in E. \qquad (14)$$

The paths from the root to each of the other nodes in the SPT are connected i.e. for each node *i* (excluding the root and last node *k*) on the path $P_{r,k}$, if one of the incoming edges to node *i* is on the path $P_{r,k}$ (i.e. $v_{j,i,k}$=1) then one of the outgoing edges from node *i* must also be on the path $P_{r,k}$ (i.e. $v_{i,j,k}$=1). This is formulated as

$$\sum_{\forall e_{i,j} \in E^-(i)} v_{j,i,k} = \sum_{\forall e_{i,j} \in E^+(i)} v_{i,j,k} \quad \begin{array}{l} \forall k \in \{N-r\}, \\ \forall i \in \{N-\{k,r\}\} \end{array} . \quad (15)$$

There is only one incoming edge to the last node k of path $P_{r,k}$ and no outgoing edge from node k. Therefore

$$v_{i,j,j} = y_{i,j} \qquad \forall e_{i,j} \in E . \quad (16)$$

Using variable $v_{i,j,k}$, we are able to formulate the weight of path from the root to each of other nodes in the graph G as

$$\mathrm{w}(P_{r,k}) = \sum_{\forall e_{i,j} \in E} v_{i,j,k} a_{i,j} \quad \forall k \in \{N-r\} \ . \quad (17)$$

The objective criterion (1) minimizes the maximal weight of path $w(P_{r,k})$. We add the new variable z determining the maximum of the weights of the paths and this variable z is then used in the objective criterion. Therefore we minimize the value of z constrained by $n-1$ inequalities

$$z \ge \mathrm{w}(P_{r,k}) = \sum_{\forall e_{i,j} \in E} v_{i,j,k} a_{i,j} \quad \forall k \in \{N-r\} . \quad (18)$$

The degree-bounded shortest path tree (db-SPT) problem is formulated as an optimization problem in the form of an integer linear programming (ILP), as summarized in Figure 6.

min z

subject to:

$$\sum_{\forall e_{i,j} \in E^+(i)} x_{i,j} + 1 = \sum_{\forall e_{j,i} \in E^-(i)} x_{j,i} \qquad \forall i \in \{N-r\}$$

$$\sum_{\forall e_{i,j} \in E^+(r)} x_{r,j} = n - 1$$

$$x_{i,j} \ge y_{i,j} \qquad \forall e_{i,j} \in E$$

$$\frac{x_{i,j}}{n-1} \le y_{i,j} \qquad \forall e_{i,j} \in E$$

$$\sum_{\forall e_{i,j} \in E^+(i)} y_{i,j} \le outDb(i) \qquad \forall i \in N$$

$$\sum_{\forall e_{i,j} \in E} y_{i,j} = n - 1$$

$$v_{i,j,j} = y_{i,j} \qquad \forall e_{i,j} \in E$$

$$\sum_{k \in \{N-r\}} v_{i,j,k} = x_{i,j} \qquad \forall e_{i,j} \in E$$

$$\sum_{\forall e_{i,j} \in E^-(i)} v_{j,i,k} = \sum_{\forall e_{i,j} \in E^+(i)} v_{i,j,k} \quad \begin{array}{l} \forall k \in \{N-r\}, \\ \forall i \in \{N-\{k,r\}\} \end{array}$$

$$z \ge \sum_{\forall e_{i,j} \in E} v_{i,j,k} a_{i,j} \qquad \forall k \in \{N-r\}$$

$$x_{i,j} \ge 0 \qquad \forall e_{i,j} \in E$$

where $y_{i,j}$ and $v_{i,j,k}$ are binary variables.

Figure 6 : ILP formulation of the degree-bounded shortest path tree (db-SPT) problem.

The number of constrains and variables depends on n (the number of nodes) and on m (the number of edges) in a given $G(N,E)$, the reduced graph representing the overlay network of hosts. The SPT problem in Figure 4 has m variables ($x_{i,j}$) and $n+m$ constraints. The db-SPT problem in Figure 6 has $(n+1)m+1$ variables ($x_{i,j}$, $y_{i,j}$, $v_{i,j,k}$ and z), where nm variables are integer and n^2+5m+2 constraints.

5. Experimental results

We used the GLPK (GNU Linear Programming Kit) solver [5] and its Matlab MEX interface (GLPKMEX [6]) to solve the integer linear programming formulation of the degree-bounded shortest path tree problem (Figure 6). GLPK is a simplex-based solver able to handle LP problems with up to 100 000 constraints. The build-in ILP solver is based on the branch and bound method (the LP relaxation of the ILP problem is solved by the simplex method and then the branch and bound method searches for the integer solution) and it is able to solve problems up to 300 integer variables. This computational restriction limits the maximal number of nodes in the graph representation of the overlay multicast network. In the graph without self-loops with n nodes the maximal number of edges is $n(n-1)$. For the graph with 7 nodes ($n=7$) and maximal number of edges ($m=7\cdot6=42$) the ILP formulation of corresponding db-SPT problem has 294 ($nm=7\cdot42$) integer variables i.e. the upper limit of the number of integer variables of the GLPK solver. Therefore GLPK solver can rationally solve db-SPT problems with at most 7 nodes.

5.1. Experiments

The first experiment shows the complexity of the db-SPT problem depending on the number of nodes and edges in corresponding overlay multicast network. One hundred random instances of an overlay multicast network topology were generated for each number of nodes n and edges m. The random generator used discrete uniform distribution U on the interval $(0, n)$. For each node from the set {5, 7, 8} the number of edges was set to 50%, 75% and 100% of the maximal number of edges ($n(n-1)$) in the graph with n nodes. The db-SPT problem with 10 nodes is added only for the comparison. The out-degree bounds (*outDb(i)*) for each instances were chosen from a discrete uniform distribution on the interval *(0,n)*, and *(0,n/2)*. The interval *(n,n)* represents the instances without out-degree bounds (*outDb(i)=n*). Consequently the ILP solver of GLPK was used to solve these random generated db-SPT problems. The task solving one instance of db-SPT problem was interrupted and labelled as incomplete when a computing time exceeded the upper time limit of 300 s. Then the efficiency represents the percentage of the completed tasks. The efficiency, average number of processed sub-problems from branch and bound method used in ILP solver of GLPK and average CPU time for each

		outDb(i)=n			outDb(i)∈ U(0,n)			outDb(i)∈ U(0,n/2)		
n	m	efficiency [%]	# sub-problems	time [s]	efficiency [%]	# sub-problems	time [s]	efficiency [%]	# sub-problems	time [s]
5	10 (50%)	*10*	*7*	-	*100*	*7*	-	*100*	*6*	-
	15 (75%)	*100*	*14*	-	*100*	*43*	-	*100*	*64*	-
	20 (100%)	*100*	*24*	-	*100*	*131*	-	*100*	*160*	-
7	21 (50%)	*100*	*237*	*0.32*	*100*	*254*	*0.37*	*100*	*291*	*0.48*
	32 (75%)	*100*	*1245*	*3.17*	*99*	*1223*	*3.66*	*100*	*1312*	*4.11*
	42 (100%)	*98*	*1747*	*6.74*	*98*	*3205*	*14*	*100*	*3633*	*17.3*
8	28 (50%)	*100*	*872*	*2.2*	*100*	*1166*	*3.3*	*100*	*1493*	*4.7*
	42 (75%)	*100*	*3694*	*18*	*100*	*3657*	*20.2*	*93*	*4667*	*30.3*
	56 (100%)	*100*	*6763*	*51.4*	*100*	*5556*	*48.5*	*85*	*6633*	*67.6*
10	45 (50%)	-	-	-	*60*	*7761*	*110*	-	-	-
	68 (75%)	-	-	-	*53*	*8794*	*147*	-	-	-
	90 (100%)	-	-	-	*30*	*9652*	*149*	-	-	-

Table 1: The time complexity of the db-SPT problems formulated in ILP framework and solved by the ILP solver. The variables *n* and *m* represent the number of nodes and edges, respectively. The value in the parenthesis behind the number of edges represents the percentage of the maximal number of edges (*n(n-1)*) for the graph with *n* nodes. One hundred of the instances were generated for each number of nodes and edges. The *efficiency* represents the percentage of completed tasks accomplished in 300 s. *#sub-problems* represents the mean value of number of processed sub-problems from branch and bound method of ILP solver and *time* represents the mean value of time of task solving one instance of db-SPT problem.

instance of appropriate db-SPT problem are shown in Table 1.

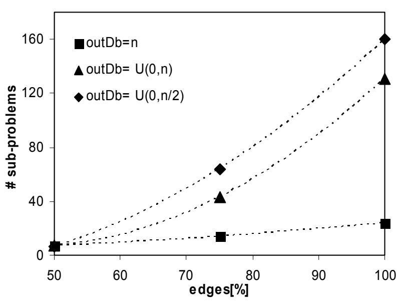

Figure 7: The db-SPT problem with 5 nodes (*n=5*) and three different ranges of *outDb(i)*.

The efficiency is in most of cases equal to 100% i.e. all instances of db-SPT problems are solved till time limit. The db-SPT problem with 10 nodes and more than 45 edges has more than 450 integer variables (10·45) thereby it exceeds the usual limit of integer variables of the ILP solver of GLPK and therefore the efficiency rapidly decrease.

The complexity (i.e. the number of processed sub-problem and CPU time) increases with increasing number of nodes and edges and with decreasing range of out-degree bounds (outDb(i)). This trend is shown in Figure 7 and Figure 8, where the X axis states for the number of edges equal to 50%, 75% and 100% of the maximal number of edges in the graph with *n* nodes and the Y axis states for the number of processed sub-problems.

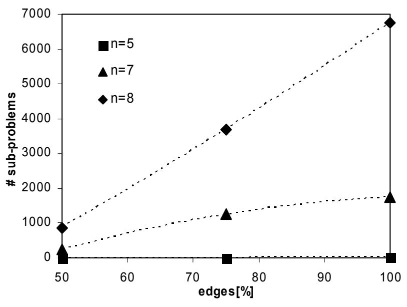

Figure 8: The db-SPT problem with 5, 7 and 8 nodes and fixed *outDb(i)* equal to n.

5.2. Comparison to B&B algorithm

The ILP solutions of randomly generated db-SPT problems are compared with the results of the Branch and bound algorithm (B&B) presented in [9]. B&B algorithm is one of the enumeration methods considering certain solutions only indirectly, without actually evaluating them explicitly. As its name implies, the B&B method consists of two fundamental procedures: branching and bounding. Branching is the procedure of partitioning a large problem into two or more mutually exclusive sub-problems. Furthermore the sub-problems can be partitioned in a similar way, etc. Bounding calculates a lower bound on the optimal solution value for each sub-problem generated in the branching process while using polynomial techniques such as Floyd's algorithm.

The second experiment compares the complexity of the ILP algorithm with the B&B algorithm. The time parameter depends on the algorithm implementation and the computing capacity. Therefore we have to use an independent and common parameter for both ILP and B&B methods. The ILP solver of GLPK uses also a branch and bound method and therefore we can use the number of processed sub-problem i.e. the number of inspected vertices in the search tree, as a common parameter for the comparison between the ILP algorithm and B&B method as shown in Table 2. The instances of above generated db-SPT problems are used in this experiment too.

		ILP		B&B	
		# sub-problems		# inspected vertices	
n	m	outDb(i)=		outDb(i)=	
		n	n/2	n	n/2
5	10	*1*	*1*	*11*	*12*
	20	*1*	*273*	*28*	*33*
7	21	*353*	*275*	*51*	*53*
	42	*600*	*1004*	*313*	*312*
8	28	*1244*	*1231*	*148*	*148*
	56	*1630*	-	*551*	*498*

Table 2: The complexity of the ILP algorithm and B&B method. The meaning of the number of processed sub-problems from branch and bound method of ILP solver is the same as the number of inspected vertices in the search tree of B&B method.

If the number of processed sub-problems is equal to 1 then the solution of the LP relaxation of the db-SPT problem is integer and branch and bound method of ILP solver only verifies this result.

The algorithms were tested on PC Pentium 4 (2.4GHz) with 256 MB RAM and MS Windows XP OS.

6. Conclusion and future work

We have shown that construction of time optimal multicast tree is easily formulated as SPT problem, the classical problem of the graph theory. This paper presents the approach to this problem based on the linear programming framework. The SPT problem is extended by a possibility to control the load of the network (i.e. routers) thereby it forms new degree-bounded SPT problem. This problem is defined and solved by integer linear programming. The results can be used in the multicast communication network for building/reshaping a multicast distribution tree.

The faster time responses should be achieved by the commercial ILOG CPLEX solver instead of the current using GNU GLPK solver. On very large scale instances CPLEX dual simplex algorithm is 10 - 100 times faster than the GLPK simplex solver and, of course, much more robust. In a future work, we will try to find approximation algorithm (i.e. the algorithm finding non optimal solution with bounded difference between the objective value of this solution and objective value of optimal solution) solved in polynomial time.

The designed optimization algorithm is centralized. Disadvantages of a centralized realization are the possibility of failure, high computing and communication load of the source and requirement of the knowledge of a network topology in the source. Possible solution is a distributed algorithm solving the degree-bounded shortest path tree problem which is also the objective of our future work.

Finally it is needed to mention that the ILP algorithm given in this paper could be applied in the reshaping procedure of the distributed algorithm maintaining the multicast tree of reasonably small size. Reshaping procedure can monitor the traffic of the multicast communication during a normal run. It can make additional measurements of RTT especially for critical nodes and it can continuously gather RTT values from all nodes to the source node. Then the reshaping procedure can completely recalculate the tree locally in the source whenever it is needed.

References

[1] R. Diestel, *Graph Theory*, Springer-Verlang, New York, 2000.

[2] K.R. Ahuja, L.T. Magnati, B.J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice hall, 1993.

[3] M. Gibs, *Introduction to Multicast Networking*, River Stone networking - Advanced technical documentation, 2000.

[4] J. Moy, RFC 2328 - OSPF Version 2, Ascend Communications, Inc., 1998.

[5] A. Makhorin, *GLPK 4.8,* Moscow Aviation Institute, Russia, http://www.gnu.org/software/glpk/glpk.html.

[6] N. Gieorgetti, *GLPKMEX,* University of Siena, Italy, http://www-dii.ing.unisi.it/~giorgetti/downloads.html.

[7] Ch. A. Papadimitriou, *Computational Complexity*, Addision-Wesley Publishing Company, Inc., 1995.

[8] Y.-H. Chu, S. G. Rao, and H. Zhang, *A case for end system multicast*, In *ACM SIGMETRIC*S. ACM, 2000.

[9] 0. Dolejš, Z. Hanzálek, "Optimality of the Tree Building Control Protocol", In: *International Conference on Parallel and Distributed Processing Techniques and Applications.* Las Vegas: CSREA Press, 2002, vol. 4, p. 1685-1686. ISBN 1-892512-90-4.

[10] L. Mathy, R. Canonico, D. Hutchison, "An Overlay Tree Building Control Protocol", In proceedings of the *3rd International COST 264 Workshop on Networked Group Communication (NGC)*, London, UK, November 2001, LNCS 2233, Springer-Verlag.

[11] S. Pallottino, M.G. Scutella, "Dual algorithms for the shortest path tree problem", *Networks 29*, 125-133.

[12] P. Narvaez, K.-Y. Siu, H.-Y. Tzeng, "New Dynamic SPT Algorithm based on a Ball-and-string Model", In proceedings of the *IEEE Infocom*, 1999.

[13] D. Pendarakis, S. Shi, D. Verma, " ALMI: An Application Level Multicast Infrastructure", In proceedings of *3rd Usenix Symposium on Internet Technologies & Systems*, March 2001.