Optimal configuration of an optical network providing predefined multicast transmissions

Vincent Reinhard a,b, Johanne Cohen b, Joanna Tomasik a,* , Dominique Barth b, Marc-Antoine Weisser a

a SUPELEC Systems Sciences, Computer Science Dpt., 91192 Gif sur Yvette, France
b PRiSM, University of Versailles, 45 avenue des Etats-Unis, 78035 Versailles, France

Abstract

Operators of networks covering large areas are confronted with demands from some of their customers who are virtual service providers. These providers may call for the connectivity service which fulfills the specificity of their services, for instance a multicast transmission with allocated bandwidth. On the other hand, network operators want to make profit by trading the connectivity service of requested quality to their customers and to limit their infrastructure investments (or do not invest anything at all).

We focus on circuit switching optical networks and work on repetitive multicast demands whose source and destinations are à priori known by an operator. He may therefore have corresponding trees “ready to be allocated” and adapt his network infrastructure according to these recurrent transmissions. This adjustment consists in setting available branching routers in the selected nodes of a predefined tree. The branching nodes are opto-electronic nodes which are able to duplicate data and retransmit it in several directions. These nodes are, however, more expensive and more energy consuming than transparent ones.

In this paper we are interested in the choice of nodes of a multicast tree where the limited number of branching routers should be located in order to minimize the amount of required bandwidth. After formally stating the problem we solve it by proposing a polynomial algorithm whose optimality we prove. We perform exhaustive computations to show an operator gain obtained by using our algorithm. These computations are made for different methods of the multicast tree construction. We conclude by giving dimensioning guidelines and outline our further work.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Optical networks have become a dominant technology in modern networks covering large areas. Their advantage consists in providing an ultra-high bit rate obtained with slight energy consumption. All-optical networks are particularly interesting from economic and ecological point of view because a cost of transparent routers is low and their energy consumption is negligible [1].

Modern networks face a growing demand on the part of service providers. New offered services are more complex than the simple connectivity service assured traditionally by network operators. Providers sell services like storage and computation together with connectivity service to their customers. The part of this market ensuring on-the-fly resource allocation, called for commercial reasons Cloud Computing [2], is under a rapid development. In
order to meet the demands of their customers, virtual service providers have to purchase a guaranteed connectivity service at network operators. At the same time, network operators can deal with numerous virtual service providers. They are interested in using their network resources most efficiently and in this way minimize the cost of a prospective extension of their existing infrastructure.

We studied the mechanisms to execute distributed applications in an optical mesh network in the context of the CARRIOCAS project [3,4]. Unlike a customary approach applied in grids where applications benefit from a dedicated network infrastructure [5], this project went into the study of the coexistence of massive distributed applications in a network whose operator should make financial profit. With GMPLS [6] deployed, the CARRIOCAS network has to ensure both unicast and multicast transmissions. Routers which are able to duplicate data and send it in several directions allow a network operator to lower the bandwidth amount necessary to construct a multicast tree. On the other hand, these branching nodes are more expensive and more energy consuming than the transparent ones. The realistic assumption is thus that only a subset of routers is equipped with the duplicating functionality. In [7] we presented our solution to the problem consisting in the construction of a tree to any multicast request with minimization of the amount of used bandwidth under assumption of a limited number of branching nodes whose locations have already been set. The solution is heuristic because we proved that this problem is NP-complete and inapproximable. It turned out to be the most effective when the branching nodes were placed in the most homogeneous way in a network. The most homogeneous placement of $k$ branching nodes represents in fact a solution to the $k$-center problem which is also NP-complete [8].

Our study mentioned above inspired us to explore certain special cases of multicast demands in circuit switching networks. A network operator can know in advance recurrent multicast transmissions which require a lot of bandwidth (as video-on-demand applications, for instance). Being aware of frequent demands for identical (or almost identical) multicast transmissions an operator may have corresponding trees “ready to be allocated” and adapt his network infrastructure according to these recurrent transmissions. This adjustment may consist in setting available branching routers in the selected nodes of the predefined tree. In this paper we are interested in the choice of nodes of a multicast tree where the branching routers should be located in order to minimize the amount of required bandwidth. This approach allows an operator to make his network more efficient without any additional cost.

In the following section we make a survey of existing solutions to multicast tree allocation and explain the specificity of branching routers. In Section 3 our problem is stated in the formal way. We also formulate (Section 4) the solution properties. Next, we propose an algorithm to solve our problem, compute its complexity, and prove that it gives an optimal solution. Our problem is evidenced to be polynomial. Section 6 presents the results of bandwidth requirements for multicast trees depending on the number of available branching routers. The multicast trees which are subject of this analysis have been obtained by two methods, the first one based on the shortest path approach and the second one based on the Steiner tree approach. In the final section we give the conclusions and outline our further work.

2. Multicast tree construction

There are several schemes for multicasting data in networks [9,10]. We present here the schemes adapted to optical circuit switching networks. The first one is to construct virtual circuits from the multicast source to each destination. Such a scheme is equivalent to multiple unicasts (Fig. 1) and the network bandwidth used by a large multicast group may become unacceptable [11].

In another scheme the multicast source sends data to the first destination and each destination acts as a source for the next destination until all destinations receive the data flow. In yet another scheme, intermediate routers make copies of data packets and dispatch them to their successors in the multicast tree. This solution allows the multicast transmission to share bandwidth on the common links. Numerous multicast tree algorithms, which follow the latter scheme, have been proposed and can roughly be classified into two categories [9]. The first category contains the algorithms based on the shortest path while minimizing the weight of the path from the multicast source to each destination. The second category contains algorithms based on the Steiner tree problem [12–15] which we formally define in Section 3. The algorithms derived from the Steiner tree problem minimize the total weight of the multicast tree. They are heuristic because the Steiner tree problem is NP-complete [14].

From the technological point of view, routers able to duplicate packets introduce a supplementary delay due to O/E/O conversions and are more expensive. For these reasons network operators want to limit the number of such routers which we call “diffusing nodes” or “branching nodes”. The diffusing nodes which we consider are not equipped with the functionality “drop-and-continue” [16]. The “drop-and-continue” operation mode, which allows a node to retrieve optical packets and send them ahead to the next router, is nowadays applied in practice exclusively in border routers. The fact, that the packet replication is performed on the electronic level, allows us
A multicast request is a pair \((A, R)\) which models two communication directions between two vertices. \(A\) is the multicast source, \(R \subseteq V\) is the set of multicast destinations. We suppose that all multicast requests which we deal with can be transmitted in the network as a set of unicast transmissions (see Section 2). Therefore, we do not have to make precise the amount of data to transfer. For a given multicast request \(\epsilon\) we first determine its tree, \(A_\epsilon = (V_{A_\epsilon}, E_{A_\epsilon})\). This tree is a subgraph of \(G\) rooted in \(e\), whose leaves are in the set \(R\) and whose arcs are directed from the root towards the leaves. We note \(D_{A_\epsilon}\) the diffusing nodes in \(A_\epsilon\). We assume that there are \(k\) diffusing nodes, \(|D_{A_\epsilon}| = k\). We now try to determine the choice of diffusing nodes in order to minimize the bandwidth consumption.

We will adopt as a metric of the bandwidth used by a multicast request a total number of arcs which construct its tree. We take into account the fact that an arc may pass through \(u\) or which terminate in \(u\).

- the origin of a path in \(S(D_{A_\epsilon})\) is either \(e\) or a node of \(D_{A_\epsilon}\); in the latter case, the node of \(D_{A_\epsilon}\) is also the final extremity of a path in \(S(D_{A_\epsilon})\).
- any node of \(a \in D_{A_\epsilon}\) is in a path \(p \in S(D_{A_\epsilon})\) only if it is the final extremity or the origin of \(p\).

The metric load of \(A_\epsilon\) is defined as a sum of lengths of all paths in \(S(D_{A_\epsilon})\). The optimization problem which consists in placing \(k\) diffusing nodes can be thus formalized as:

**Definition 1.** (Diffusing Nodes in Multicast Tree Problem (DNMTP)).
**Data:** A directed connected symmetrical graph \(G = (V,E)\), a multicast request \(\epsilon = (e,R)\), a rooted multicast tree corresponding to this request \(A_\epsilon\), and a natural number \(k\).
**Goal:** Find \(D_{A_\epsilon} = \{a_1, \ldots, a_k\}\) so that load of \(A_\epsilon\) is minimal.

### 3. Formalization of optimization problem

An optical network is modeled by a directed connected symmetrical graph \(G = (V,E)\) \cite{17}. The symmetry property models two communication directions between two vertices. A multicast request is a pair \(\epsilon = (e,R)\), where \(e \in V\) is a multicast source and \(R \subseteq V\) is a set of multicast destinations. We suppose that all multicast requests which we deal with can be transmitted in the network as a set of unicast transmissions (see Section 2). Therefore, we do not have to make precise the amount of data to transfer. For a given multicast request \(\epsilon\) we first determine its tree, \(A_\epsilon = (V_{A_\epsilon}, E_{A_\epsilon})\). This tree is a subgraph of \(G\) rooted in \(e\), whose leaves are in the set \(R\) and whose arcs are directed from the root towards the leaves. We note \(D_{A_\epsilon}\) the diffusing nodes in \(A_\epsilon\). We assume that there are \(k\) diffusing nodes, \(|D_{A_\epsilon}| = k\). We now try to determine the choice of diffusing nodes in order to minimize the bandwidth consumption.

We will adopt as a metric of the bandwidth used by a multicast request a total number of arcs which construct its tree. We take into account the fact that an arc may transport the same data more than once. To define this metric formally we start by determining the situations in which a request \(\epsilon\) is satisfied by a set of paths in the multicast tree, \(S(D_{A_\epsilon})\). These situations are as follows:

- every node of \(R\) is the final extremity of exactly one path in \(S(D_{A_\epsilon})\);
- every node of \(D_{A_\epsilon}\) is the final extremity of at most one path in \(S(D_{A_\epsilon})\).

### 4. Properties of the solution induced by a set of diffusing nodes

We now focus on a given multicast \(\epsilon\) and we omit the subscript \(\epsilon\) in the formulæ for their clarity, i.e. \(A_\epsilon = (V_{A_\epsilon}, E_{A_\epsilon})\) will be noted form now onwards as \(A\). This section is dedicated to studying properties of the solution induced by a set \(D\) of diffusing nodes, \(S(D)\). We introduce the notation used for its description and illustrate it in Fig. 3. For any \(u, u \in V\) in \(A\) we define \(A^u\) as a sub-tree of \(A\) rooted in \(u\). We also define three parameters of \(u\) in \(A\). A set \(D^u\) is a set of diffusing nodes in tree \(A^u\), \(D^u \subseteq D\). A set \(R^u\) is a set of destinations nodes in tree \(A^u\), \(R^u \subseteq R\). \(a^u\) is the arc connecting \(A^u\) from the remainder of \(A\).

**Definition 2.** Let \(D\) be a set of vertices in \(A\), \(D \subseteq V_{A}\). Let \(u\) be a vertex in \(A\), \(u \in V_{A}\). The path number, \(pn(u)\), is a number of paths in a solution \(S(D)\) spanned on \(A\) which pass through \(u\) or which terminate in \(u\).
Definition 3. The window of the solution $S(D)$ on arc $a^u$, $w(a^u, D)$, is an triplet of integers ($\beta, d, \text{load}$) where $\beta$ is its path number $\text{pn}(u)$, $d = |D^0|$ and load represents the load of $S(D)$ in tree $A^u$, $\text{load} = \text{load}_D$.

We can notice that each solution induced by the set of diffusing nodes $D$, can be defined by each window for each arc of tree $A$. From our point of view the fact that $w(a^u, D)$ can be computed from the windows of the successors of $u$ is crucial. The following lemma gives the window computation scheme for the situations in which $u$ has a single successor.

Lemma 1. Let $D$ be a set of vertices in $A$, $D \subseteq V_A$. Let $u$ be a node having a single successor $u_1$ of $A$. Window $w(a^u, D)$ is equal to
\[
\begin{cases}
(1, d + 1, \text{load} + 1) & \text{if } u \in D, \\
(b + 1, d, \text{load} + b + 1) & \text{if } u \notin D \text{ and } u \in R, \\
(b, d, \text{load} + b) & \text{if } u \notin D \text{ and } u \notin R,
\end{cases}
\]
where the window on arc $a^{u_1}$ is $(b, d, \text{load})$.

Proof. First, we assume that $u$ is a diffusing node, $u \in D$. By definition of path number (Def. 2), node $u$ has a path number equal to one. Since $w(a^u, D) = (b, d, \text{load})$, tree $A^u$ contains one more diffusing node than tree $A^{u_1}$.

Second, we assume that $u$ is not a diffusing node and tree $A^u$ contains exactly the same set of diffusing nodes as tree $A^{u_1}$. If $u$ is a destination, $u \in R$, then $u$ is the final extremity of exactly one path. Consequently, the path number for vertex $u$ is equal to the path number for vertex $u_1$ plus one. If $u$ is intermediate node, $u \notin R$, then the path number for vertex $u$ is equal to the path number for vertex $u_1$.

From these statements, we can compute the load of the solution $S$ induced by $D$ in tree $A^u$. The load increases by the path number for vertex $u$. \qed

Using the same arguments as in the proof of Lemma 1, we extend it when $u$ has several successors.

Lemma 2. Let $u$ be a node of $A$ having $\ell$ successors $u_1, \ldots, u_\ell$. Window $w(a^u, D)$ is equal to
\[
\begin{cases}
(1, d' + 1, \text{load}' + 1) & \text{if } u \in D, \\
(b' + 1, d', \text{load}' + b' + 1) & \text{if } u \notin D \text{ and } u \in R, \\
(b', d', \text{load}' + b') & \text{if } u \notin D \text{ and } u \notin R,
\end{cases}
\]
where the window on arc $a^{u_i}$ is $(b_i, d_i, \text{load}_i)$ for any $i$, $1 \leq i \leq \ell$.

Now, we want to compare two solutions by introducing a partial order $\preceq_S$ for each node $v \in V_A$. The order $\preceq_S$ is introduced by ordering each coordinate of window triplets according to the “standard” integer order $\leq$.

Definition 4. Let $D$ and $D'$ be two subsets of vertices in $A$. Let $v$ be a vertex and $w(a^v, D) = (b, d, \text{load})$, $w(a^v, D') = (b', d', \text{load}')$. $S(D) \preceq_S S(D')$ if and only if the following three conditions are simultaneously satisfied: (i) $b \leq b'$, (ii) $d \leq d'$, (iii) $\text{load} \leq \text{load}'$.

This order gives the possibility to improve a solution $S(D)$ by finding a better set of diffusing nodes.

Proposition 1. Let $u$ be a vertex in $A$. Let $D$ and $D'$ be two subsets of vertices in $A$ such that $S(D) \preceq_S S(D')$. Then the solution induced by $D'$, where $D' = (D \setminus D^0) \cup D^0$, satisfies the following property:
\[S(D') \preceq_S S(D')\] for all nodes $v$ not in $A^u$.

Proof. First, we focus on vertices $v$ outside $A^u$ and not in $P$. Since $D' = D^0$, arc $a^u$ has the same window of the solution induced by $D$ and of the solution induced by $D'$, $w(a^u, D) = w(a^u, D')$.

Second, we focus on vertices $v$ which are in $P$. By definition of the partial order $\preceq_S$ (Def. 4), if $S(D) \preceq_S S(D)$ then for the windows $w(a^v, D) = (b, d, \text{load})$ and $w(a^v, D') = (b', d', \text{load}')$, we have (i) $b \leq b'$, (ii) $d \leq d'$, and (iii) $\text{load} \leq \text{load}'$. Now, we can compute $w(a^v, D')$, where $t$ is the predecessor of node $u$. Let $(b_1, d_1, \text{load}_1)$ be the window of the solution $S(D) \preceq_S S(D')$.

From Lemma 2, if $t \in D'$, then $w(a^v, D')$ is $(1, d_1 + d + \text{load}_1)$, and $w(a^v, D') = (b_1, d_1, \text{load}_1)$. Thus $S(D') \preceq_S S(D')$. We can apply the same arguments as previously for the other case. The same reasoning goes for each vertex of this path starting from the predecessor of $t$ until the root. This completes the proof of Proposition 1. \qed

Next, we define a notion of the optimality of $D$ in sub-tree $A^u$ of $A$.

Definition 5. Let $u$ be a vertex in $A$, $u \in V_A$. Let $D$ be a subset of vertices in $A$, $D \subseteq V_A$, $D$ is sub-optimal for node $u$ if and only if for any $D'$ which is a subset of vertices in $A$ such that $d = d'$ and $b = b'$, we have $\text{load} \leq \text{load}'$, where $w(a^u, D) = (b, d, \text{load})$ and $w(a^u, D') = (b', d', \text{load}')$.

In other words, a solution $D$ is sub-optimal for node $u$ if and only if it is optimal in sub-tree $A^u$ of $A$.

The following statement expresses the property that a set of diffusing nodes which is optimal in $A^u$ is also optimal for all sub-trees rooted in $u$ successors.

Proposition 2. Let $u$ be a vertex of $A$ having $\ell$ successors $u_1, \ldots, u_\ell$. Let $D$ be a subset of vertices in $A$. If $D$ is sub-optimal for node $u$, then $D$ is also sub-optimal for node $u_i$ for any integer $i$, $1 \leq i \leq \ell$.

Proof. We can prove this proposition by contradiction. Assume that there is at least one successor $u_i$ of $u$ such that $D$ is not sub-optimal for node $u_i$. So it implies that there exists a subset $D'$ such that $D'$ is sub-optimal for node $u_i$ and such that $d = d'$ and $b = b'$, we have $\text{load} \leq \text{load}'$ where $(b, d, \text{load})$ is window $w(a^u, D)$ and $(b', d', \text{load}')$ is $w(a^u, D')$. Consequently, using Lemma 2, we can construct a subset $D''$ such that $D'' = (D \setminus D^0) \cup D^0$ and such that $S(D'') \preceq_S S(D')$. This implies that $D$ is not optimal. So, there is a contradiction. \qed
5. Algorithm, its complexity and optimality

Our algorithm is based on the bottom-up dynamic approach. We resume the notation used for its description. For any $u, v \in V_{D}$ in $A$ we define $A^u$ as a sub-tree of $A$ rooted in $u$. We also define two parameters of $v$ in $A$.

The height $h(u)$ is a distance between $u$ and multicast source $e$ in $A$. We also note $h_{\max} = \max_{v \in V_{D}} h(u)$. The path number $pn(u)$ is a number of paths in a solution $S(D)$ with a given set of diffusing nodes, $D_{\max}$, spanned on $A$ which pass through $u$ or which terminate in $u$.

As the minimal load of the sub-optimal set for it should be in the leaves of $A$. As we have said above, our algorithm to solve the DNMTP (Definition 1) attributes to each node $u$ its $M(u)$ and $L(u)$ starting from the leaves whose height is $H = h_{\max}$ and performing the bottom-up scheme with $H = H - 1$ until the root $e$ is reached ($H = 0$). The attribution of $M(u)$ and $L(u)$ to $u$ is realized by the procedure Mat_VEC_Filling (Fig. 4). This procedure takes a node $u$ and its corresponding sub-tree $A^u$ as data. Intuitively speaking, this is a modified breadth-first search [18] in which one arbitrarily chosen successor, treated first, computes its matrix and vector (the First_Succ_Mat_VEC procedure, Fig. 5) in a different way from its siblings (the Others_Succ_Mat_VEC procedure, Fig. 6). The leaves have the “unitary” matrix and vector assigned.

The procedure First_Succ_Mat_VEC (Fig. 5) operates on a node $u$ and one of its successors $v$ for which $M(v)$ and $L(v)$ are already known as Mat_VEC_Filling follows a bottom-up approach. It uses the variable $elT$ to store the non-zero elements in a column $i$ of $M(u)$ and $L(u)$.

The procedure executes the function $\min^+$ whose two arguments are natural. It returns a minimum of these two values in exception of the case in which one of the arguments is zero. The other positive argument is when returned.

The main idea is based on the observation that the weight of the multicast tree in $A^u$ is equal to the multicast weight in $A^u$ incremented by the weight of reaching $u$ which is itself equal to path number $pn(u)$. Let us remind the reader that $pn(u) = 1$ when $u$ is a diffusing node and $pn(u)$ is a row index of matrix $M(u)$ corresponding to node $u$ which is not diffusing.

Remark 1. From Lemma 1 and Proposition 1, we can deduce, that if $u$ has one successor $v$ for any $ij$ such that $0 \leq i < k$ and $1 \leq j < |R|$: $L(u) = 1 + \min^*(M_0(v):j:1 \leq j \leq |R|)$.

The procedure First_Succ_Mat_VEC computes the formulae here above.

In lines 1–4 $L(u)$ is computed for $u$ seen as a diffusing node. On the $i$th step the smallest positive weight is chosen between weights of its predecessor $v$ seen as a diffusing and a non-diffusing node. These weights are taken for $v$
with one diffusing node less because $u$ itself is diffusing. This weight is increased by the weight of reaching $u$ which is one as $u$ is diffusing. Lines 5–9 fill up $M(u)$ when $u$ is seen as non-diffusing. Line 7 treats the case in which only one path passes through or terminates in $u$. The successor of $u$ can be either a diffusing or non-diffusing node. As $L_0(v)$ does not exist, we use $L_0(v) = 0$ in numerical computations. Otherwise (line 8) its successor has to be a non-diffusing node. The case in which $u$ is a destination despite the fact that it is not a leaf in $A$ is treated in lines 10–12 as the weight of the access to $u$ has to be added.

The procedure $\text{Others\_Succ\_Mat\_Vec}$ (Fig. 6) operates on a node $u$ and its successors $v$ different from $u$ which has already been examined in $\text{First\_Succ\_Mat\_Vec}$. The procedure uses the variable $e_{VT}$ as the procedure $\text{First\_Succ\_Mat\_Vec}$ does. Furthermore, the procedure makes use of the auxiliary variables $M'(u)$ and $L'(u)$ to store the new values of $M(u)$ and $L(u)$ as the current elements of $M(u)$ and $L(u)$ are still in use. The procedure $\text{Others\_Succ\_Mat\_Vec}$ is built up on the same principle as the previous one. Lines 1–11 treat the filling up of $L(u)$ and lines 12–26 treat the filling up of $M(u)$. The important difference consists in traversing all the couples $(x, y)$, $x, y = 1, 2 \ldots, k$ or $x$ or $y = 0, 1, \ldots, k$ such that $x + y = i$. It leads from the fact that this time the weight of the multicast tree in $A^u \cup A^v \cup \{u\}$ is equal to the sum of the multicast weights in $A^u \cup \{u\}$ and in $A^v$ with the branching nodes deployed in both $A^u \cup \{u\}$ and $A^v$. The matrix computation also requires an appropriate path number in order to determine the additional tree weight (lines 17–22).

**Remark 2.** From Lemma 2 and Proposition 2, we can deduce, that if $u$ has $\ell$ successors $u_1, \ldots, u_\ell$ for any $i, j, 0 \leq i < \ell$ and $1 \leq j \leq |R|$: 

- $L'_i(u) = \min \{L_{i,j}(u) + L_j(u_1), L_{i,j}(u)\} + \min_{1 \leq j \leq |R|} M_{i,j}(u) : \forall i > 0$,

- $M'_i(u) = \min \{\min_{1 \leq j \leq |R|} M_{i,j}(u) + L_{i,j}(u_1) + 1 : \forall i > 0\}$,

where $M'(u)$ and $L'(u)$ correspond to the matrix and the vector computed by the algorithm for the sub-tree of $A^u$ where $u$ has only $\ell$ successors $u_1, \ldots, u_\ell$.

The procedure $\text{Others\_Succ\_Mat\_Vec}$ computes the formulæ here above.

**Theorem 1.** The optimal set of diffusing nodes is obtained by the configuration associated to $\min_{1 \leq j \leq \ell} L_i(e)$. Its complexity is $O(k^2|R|^2|V_A|)$.

**Proof 4.** From Remarks 1 and 2, we can deduce from any $u$ in $V(A)$, the algorithm $\text{Others\_Succ\_Mat\_Vec}$ computes vector $L(u)$ in $O(k^2|R|$) (see the loops in lines 1 and 3–4 and $\min\cdot$ execution in line 7 of this algorithm in Fig. 6; we assume that a linear effort is needed to find a minimum of $|R|$ elements) and matrix $M(u)$ in $O(k^2|R|^2)$ (see lines 12, 13–14, 15 and 17–18 of this algorithm in Fig. 6) such that $\forall b, 1 \leq b \leq |R|, \forall d, 0 \leq d \leq k$, thus there are two sub-optimal sets of diffusing nodes: one has load $L_d(u)$ and the other has load $M_{d,b}(u)$.

### 6. Numerical results

Our algorithm determines the optimal localizations for $k$ diffusing nodes in a multicast tree which has already been created for a request $\epsilon = (e, R)$. As we have signaled...
in Section 2 there are numerous methods of construction of these trees. We selected two heuristic methods in order to observe their impact on the efficiency of our algorithm.

The first one establishes a shortest path (ShP) between $e$ and each $r \in R$. The corresponding multicast tree $A^\text{ShP}_e$ is a union of these shortest paths.

The second method, which is based on the 2-approximable solution of the Steiner tree problem proposed in [19], gives $A^\text{StT}$ tree. This Steiner problem formalized in terms of multicast demand can be written as:

**Definition 6** (Steiner Tree Problem (StTP)).

**Data:** A connected undirected graph $G = (V, E)$, a multicast request $\epsilon = (e, R)$, and a natural number $k$.

**Question:** Does a rooted tree $A_e$ exist such that the number of its arc is less than or equal to $k$?


These two approaches, which are notably applied in practice (see Section 2), represent two opposite visions of the optimal multicast tree placement. The ShP algorithm opts for minimization of path lengths “one by one” from the source to destinations (local perspective). The StT algorithm focuses upon minimization of used network resources (global perspective). The 2-approximable algorithm which we implemented is particularly well adapted to operate in a distributed environment [22].

To generate a graph of 200 nodes we apply the Waxman model [23] of BRITE [24] (with default parameters). We estimate with the 5% precision at the significance level $\alpha = 0.05$ the average weight of multicast tree as a function of the number of destinations for both the algorithms which construct a tree. For each number of destinations we choose uniformly in $V$ a multicast source $e$ and next, we select the destinations of this source according to the uniform distribution in $V - \{e\}$.

The measure which we are interested in is the tree weight: a number of edges used to transmit a given multicast (an edge which transports $m$ copies of a given transmission is counted $m$ times). This metric corresponds to $load_e$ (see Section 3) when the diffusing nodes are present in the network.

In order to perceive the impact of diffusing nodes on the tree weight we perform the computations with four nodes placed by our algorithm, and without them. In Fig. 7 we observe that the weight reduction obtained for ShP with the diffusing nodes is significant (about 31% for 32 destinations). The improvement obtained by the introduction of diffusing nodes into the trees built with StT (Fig. 8) is even more substantial than in the previous case (about 65% for 16 destinations). These two figures exhibit that ShP generates trees whose weight is less than those generated by StT. This fact is not astonishing as ShP always chooses a shortest path between the source and any destination.

In Fig. 9, in which the relative difference between $A^\text{ShP}_e$ and $A^\text{StT}_e$ weights as StT tree weight percentage with four diffusing nodes as a function of the number of destinations.

In order to perceive the impact of diffusing nodes on the tree weight we perform the computations with four nodes placed by our algorithm, and without them. In Fig. 7 we observe that the weight reduction obtained for ShP with the diffusing nodes is significant (about 31% for 32 destinations). The improvement obtained by the introduction of diffusing nodes into the trees built with StT (Fig. 8) is even more substantial than in the previous case (about 65% for 16 destinations). These two figures exhibit that ShP generates trees whose weight is less than those generated by StT. This fact is not astonishing as ShP always chooses a shortest path between the source and any destination.

In Fig. 9, in which the relative difference between $A^\text{ShP}_e$ and $A^\text{StT}_e$ weights as StT tree weight percentage with four diffusing nodes as a function of the number of destinations.
We now fix the number of destinations to 20 and we estimate the weights of trees obtained with ShP and StT algorithms in function of the number of branching nodes.

We remind the reader that for 20 destinations and four diffusing nodes ShP turned out to be slightly more efficient than StT. Figs. 10 and 11 also show the average weight of ShP and StT trees estimated with the absence of diffusing nodes. In accordance with the comment made above in the context of the absence of diffusing nodes, ShP trees are almost twice as good as StT ones.

The introduction of three diffusing nodes reduces the weight of $A_{ShP}^{C15}$ by about 20 additions allows one to lower the tree weight by almost 40% for 15 branching nodes. The influence of the branching nodes on the reduction of the tree weight in the StT case is striking (Fig. 11): an improvement of almost 60% in the case of three branching nodes until almost 75% for 15 of them. Confronting the results of ShP and StT with diffusing nodes we observe that StT, despite its starting point at a worse position, reaches the tree weight of 40 in the situation in which ShP has this weight of 48.
In Fig. 12 we observe the relative difference of ShP and StT tree weights for 20 multicast destinations in function of the number of diffusing nodes. It is not surprising that for this relatively large number of destinations and few diffusing nodes StT exhibits better performance than ShP. When the number of branching nodes increases and approaches the number of destinations, ShP trees become “lighter” than StT ones for the same reasons as those mentioned in the comments on Fig. 9.

The next question we ask ourselves concerns the detection of the numbers of diffusing nodes and destinations up to which StT is more advantageous than ShP. For the network investigated above the critical point is (4, 18). In Fig. 13 we mark critical points starting from which the ShP tree gives “lighter” solutions. For the points above the line we recommend ShP (for example, for three branching nodes and 30 destinations), for those below the line we recommend StT.

As the critical points depicted in Fig. 13 form a straight line whose slope is five, we are now interested in what this gradient depends on. One may guess that it is determined by the average degree of the network. Indeed, if we look at Fig. 14, the gradient decreases as the average node degree increases. Consequently, the line seen in Fig. 13 inclines with the average degree growth. Therefore we conclude that StT is more favorable for loosely connected graphs and ShP is better for dense networks.

7. Conclusions and further work

We studied a problem of infrastructural design of a commercial optical meshed network with a circuit switching routing protocol deployed. This problem was stated within the context of virtual services based on multicast transmission. It concerns frequent and voluminous multicast demands whose source and destinations are à priori known and its solution determines the localizations of branching nodes (i.e. routers with higher cost and energy consumption but which allow one to duplicate data and retransmit it in different directions). A solution to this problem allows a network operator to use its available resources more efficiently and make more profit with less, or even without any, investment.

After formally stating the problem we proposed an algorithm to solve it. Next, we proved its optimality and computed its complexity which is polynomial. We computed a gain in terms of the used bandwidth compared with multicast trees without any diffusing nodes. Among the two heuristic algorithms which we used to deploy multicast trees the first is based on the shortest past approach (ShP) and the second one exploits a solution to the Steiner tree problem (StT) in undirected graphs.

We performed exhaustive computations in order to compare the efficiency of our algorithm for multicast trees built with ShP and StT. We observed the dependency of their efficiency on the numbers of diffusing nodes and destinations. This dependency is influenced by the average network degree. StT works better in loosely connected networks whereas ShP is more efficient for strongly connected ones.

Generally speaking, we found ShP more efficient in finding a multicast tree than StT. We should not forget, however, that we used the 2-approximable algorithm. It is not excluded that a more precise StT algorithm (for example [25,26]) may give better results. We consider implementing these algorithms in order to verify their performance for our purposes.

We plan to continue this work in order to determine a specific solution in particular graphs (for example having bounded treewidth). We conjecture that our algorithm could be extended to this kind of graph. On the other hand we consider pursuing our work on optimal multicast deployment by studying the Steiner problem in certain oriented graphs.

References


Vincent Reinhard obtained his MSc diploma in Computer Science from the University of Versailles, France. He is doing his PhD thesis in Computer Science on the Quality of Service (QoS) in optical networks at the University of Versailles and SUPELEC, one of the leading French engineering schools.

Johanne Cohen is a full time researcher at CNRS. She has worked at the Versailles University since 2008. She obtained her Master degree from the Ecole Normale Supérieure de Lyon, France, in 1995 and PhD degree in Computer Science from the University Paris Sud in 1998. She was an assistant professor at the Nancy I University from 2000 to 2001 and has been a full time researcher at CNRS since 2001. She obtained her HDR (Habilitation à Diriger les Recherches) in 2009. Her research interests include algorithmics for telecommunication, graph theory, algorithmic game theory, distributed algorithmics, complexity and approximation theory.

Joanna Tomasik received her magisterium diplomas in applied mathematics (1988) and in computer science (1998) at the University of Technology in Gliwice, Poland. She obtained her PhD in 1989 from the same university. In 1988 she started to work at the Institute of Theoretical and Applied Computer Science in Gliwice. She also worked in France at the University of Versailles and the National Institute of Telecommunications and in Great Britain at the University of Edinburgh. Since 2002 she has been a professor at SUPELEC, one of the leading French engineering schools. She obtained her HDR (Habilitation à Diriger les Recherches) in 2012. Her main research domains concern performance evaluation.

Dominique Barth obtained his PhD in Computer Science (1994) at the University of Bordeaux and defended a HDR (Habilitation à Diriger les Recherches) (1999) at the University Paris-Sud. He was an assistant professor at the University Paris-Sud (1994–1999). Since 1999 he has been a professor at the University of Versailles. He created and led the Algorithmic, Combinatoric and Application team. His main research interests concern graph theory and algorithms. In the field of telecommunication networks, he focuses on QoS routing and multicast. He is and has been a leader of his team in many European and national research projects.

Marc-Antoine Weisser defended a PhD thesis in Computer Science (2007) at SUPELEC, one of the leading French engineering schools. He was a post-doctoral fellow in 2008 at the University of Versailles and a member of the Algorithmic, Combinatoric and Application team of PRiSM laboratory. Since September 2008 he has been an assistant professor in Computer Science Department of SUPELEC. His main research interests concern the graph theory and graph algorithms applied to telecommunication networks. He particularly focuses on multicast in networks in which related graph problems are variants of Steiner tree problems.