# Efficient Multicast Routing in High Speed Networks

**M. I. Smirnov**[1]
**GMD FOKUS**
**Hardenbergplatz 2**
**D-10623 Berlin, Germany**

## Abstract

The paper considers the case of multipoint routing in the high speed networking environment. Efficiency of such kind of routing, known as *multicasting* depends significantly on the quality of a network routing protocol establishing the multicast tree spanning all senders and receivers of the multicast flow. This quality in turn could be measured by minimal delay or minimal network resource consumption (minimal cost tree). The latter is known as Steiner problem in networks [1]. A new heuristic algorithm to solve the problem of shared multipoint minimal cost tree is presented together with it's formal background, cost calculation technique and estimation of needed checking overhead. The centralized version of the algorithm is demonstrated with a numerical example. The fitness of the algorithm for a distributed implementation is discussed.

## 1. Introduction

To provide an efficient way to support *one-to-many* and *many-to-many* communications in high speed networks the network layer protocols usually provide a *multicast* support. This support guarantees the effective use of the bandwidth for multi-user multimedia communications such as remote video and audio, multimedia conferencing, collaborative work, scientific visualisation and sharing of multimedia objects. The multicast support usually involves construction of a special *multicast [routing] tree (MCT)* with data packets being replicated only at splitting nodes of the MCT. Dissimilar criteria could be applied to the MCT, two major are delay and cost associated with network links [2]. This paper deals only with minimal cost criterion due to the following reasons. First, any new multicast application having bugs will be less dangerous when the MCT cost is minimized a priori. Second, any algorithm based on cost has better dynamic features (ability to redesign the MCT at run-time) which seems to have superior value for future applications [3,4]. The communication cost is a fundamental concept for routing especially when it faces new challenges imposed by high speed networks and networked applications with high resource consumption. Real costs may be changed in order to influence the network routing in some artificial way (say, trying to close some undesirable paths [2]), however, real cost considerations clearly introduced to the network mechanisms are believed to bring prerequisites for the total network performance improvement. Another associated benefit of cost employment is that of presenting a single vision of network state information to use in network control and management including interdomain issues [5]. However, communication cost is a dynamic concept which is especially important for shared MCT with a number of simultaneous senders, probably with various flow specifications. The paper addresses a new algorithm for the Shared MCT construction with minimal cost which makes use of alterations of two steps, namely *BottomUp* (looking for minimal cost chains in the whole network graph) and *TopDown* (trying to increase the so called MC-connectivity for the

1. On leave from the St. Petersburg State Electrotechnical University, St.Petersburg, Russia.

MC group members - both senders and receivers). The algorithm is a heuristic one, however analytical estimations are available for "lossed" efficiency with regard to different cost structures of the network. The algorithm tries to make cost alignment for different senders (with different flow specifications) using the same MCT. The paper is organized as follows. Section 2 depicts some preliminary considerations and discusses related work. Section 3 introduces major formalization. Section 4 gives a formal background for the algorithm, the algorithm itself with numerical example and evaluation considerations. Future work areas conclude the paper.

## 2. Problem Description and Related Work

### 2.1 Multicast Issues

The general multicast problem solution is to consider the number of issues. The heuristic design was started in concern with the following questions [1, 3 - 12]:

- Whether the MCT is to support one-to-many communication mode or many-to-many as well?
- Whether the quality of service requirements for all receivers should be unique or may differ and to what extent?
- Whether all nodes of the network could be considered as the multicast routers or only some of them (“core nodes” [6])?
- If the core nodes concept is used, then where these nodes should be located in the network in order to support as many multicast trees as possible?
- To what extent the MCT is manageable in both senses - to be able to meet dynamic Quality of Services requirements of different users and to cope with dynamic changes in the sender /receiver groups?

However when it became clear that the algorithm proposed is mostly essential for the shared MCT construction, new list of questions raised [13, also 5, 9, 14, 15]:

- Who, when and how has to calculate the limit of parallel sources for a single MCT?
- Does each broadcast source use a separate multicast trees; or does the application make use of a bidirectional multipoint tree?
- How to optimize combination of low_cost/low_delay for multiple sources?
- If a single tree is used with multiple broadcast sources, then how to make reallocation of bandwidth along the tree links (which may be necessary as the members of the connection changes)?
- If sources will send at random what is the reasonable bandwidth to be allocated for a single(shared) bidirectional MCT?

### 2.2 Communication mode

Certainly this list of questions is too large to be solved by a single algorithm, however all of the above mentioned problems are heavily influenced by the MCT topology. Particular decision could be made when the communication mode needed for particular application is known. We consider in this paper the communication mode with a set of *dedicated* senders and/or receivers which have to be connected by the MCT with different QoS requirements. At the same time when the MCT is established for all dedicated nodes any other node may join the multicast convention as a *receiver-only* offering to the multicast convention his different QoS requirements. Pre-negotiated number of senders may send in parallel with different flow specifications for each sender.

Multicast routing was studied primarily as source based multicasting. The source based multicasting is rather familiar with existing applications and, more important, with existing expectations of human behaviour during the multicast sessions. One of the most complete studies of the so called communication modes [11] addresses types of multicast based on human factors. According to their classification this paper addresses such highly dynamic communication modes as Casual and Business Meetings and use generic title *Talk Show*. The Talk Show could be described as "All participants actively send/receive data"; model of interaction - dynamic (i.e. that normally there are no restrictions imposed by the media for the data transmission); data flow is *N:N* (i.e. everyone can address to everybody, within the pre-negotiated set in case of planned session); accessibility is controlled (for business meeting) or uncontrolled (for casual meeting); event scheduling is planned (for business meeting) or unplanned (for casual meeting). As we can see from the existing most common multicasting events [16] the referred dynamic mode is not utilized and consequently the multicast infrastructure is not supporting these highly dynamic events. It is important to mention that in a TS mode we aim at MCT equal efficiency for all dedicated nodes as senders and to any other node in the network as a receiver.

### 2.3 Related Work

The multicast tree problem was intensively studied in the last decade from various viewpoints, including the MCT topology design [1, 6 9, 14, 15, 17, 18] (going to an NP-complete Steiner problem in graphs), protocol issues for different network technologies [5, 7, 8, 19 - 24], hardware design and special application problems [25 - 28] including the Internet working prototype (MBONE). In this paper we address the first issue having in mind, however the particular application communication pattern. The proposed algorithm tends to increase the *multicast connectivity* starting with a Multicast Forest (MCF), comprised by all members of the MC group. Due to the one-to-many communication pattern most of known heuristic algorithms start to construct (or to prune) the MCT from the root (sender node) to all nodes known as receivers. In [1, 9, 14, 17,18] about two dozens (including variations) of various heuristic algorithms could be found with the KMB [17] algorithm proposed by L. Kou, G. Markowski and L. Berman being one of the famous. A typical approach to analyse the quality of a heuristic is to estimate the number of it's operations and difference of obtained solution from the optimal one. This paper presents another approach: when the main part of the proposed algorithm ends with some solution which is not acceptable by the network due to it's cost then the algorithm gives possibilities to explore all "missed" opportunities which can be easily derived from the obtained MCT with the use of checking conditions presented in section 4.4. Another advantage is that no pruning is needed after receiving the solution: pruning is essential part os the main part of the algorithm. This approach seems to be practical for the MCT with high dynamics of sender/receiver groups. Yet one more consideration against any comparison of heuristics: in [18] it was shown clearly that for any pair of Steiner tree heuristics conditions could be found under which one of them beats the other and vice versa.

Normally the network is supposed to be represented with a graph with edges labelled with communication costs. Obviously the cost definition will influence greatly the final MCT, however the cost assignment problem has no unique decision [2, 29]. If the cost function is a mapping of edges into real numbers we always can encode these numbers by integers going in ascending order. We are assuming here that the communication cost is given for the default bandwidth needed for the MC session.

Due to the Talk Show communication mode and the possibility for any non-dedicated network node to join the established MCF we have to find the MCF construction algorithm with the following features. First the algorithm has to provide the equal starting position for all dedicated

nodes; in other words it has to start the MCF calculation simultaneously from all dedicated nodes. Secondly, the MCF has to utilize the less expensive parts of the rest of the network to connect every MCT into the forest. The desirable feature of the algorithm is it's scalability; we'll call the *scalability good* if the number of calculations will depend rather of the number of dedicated nodes and cost levels than the total number of nodes in the network.

## 3. Model

The network is modelled by a graph $G(V, E)$, where $V$ and $E$ denote the set of nodes and the set of interconnecting links respectively; $|V| = N$ and $|E| = M$. Links are assumed to be bidirectional and if a link is present between nodes $i$ and $j$ $(i, j \in V)$, then $e(i, j) \in E$. We assume a packet switched network in which cost of sending a packet with required quality from $i$ to $j$ is $c(i,j)$ and $c(i,j) = c(j,i)$ where $i$ and $j$ are adjacent nodes. Actually this means that if the quality is represented by needed bandwidth then the cost is to be a multiple entry. The cost for a path $(C)$ consisting of several nodes will be given as a sum of corresponding costs. And at last for every tree or mesh being a subnetwork it's cost will be given as a sum of all edge costs. For the simplicity of the rest of the paper we'll assume that all edges which doesn't correspond the current QoS specification are cut off the graph. In other words we are eliminating all infinite costs. The other assumption which simplifies the algorithm analysis without loss of generality is integer costs starting with $1$ and going to $K$.

The network is supposed to have a set of dedicated nodes $D \subset V$, where $|D| = N_d$ and every dedicated node can be a sender and a receiver of the multicast packets. The multicast forest problem consists of finding a minimum cost subgraph (forest) $F(V_f, E_f)$ spanning all dedicated nodes. $F(V_f, E_f)$ is a subgraph of $G(V, E)$ that satisfies the following conditions:

1. $\exists (F \subseteq G)$; $(|V_f| \geq N_d)$; $(D \subseteq V_f)$ ; both $F$ and $G$ are connected, i.e. both graphs has no isolated nodes;
2. $E_f \subseteq E$;
3. $C[F(V_f, E_f)] = \sum_{e(i,j) \in E_f} c(i,j) \rightarrow min$ over all subgraphs with properties 1 and 2.

## 4. The MCF Algorithm

First let us present an idea which makes the idea of the algorithm, then introduce some needed formalization and cost considerations followed by the algorithm itself. The last subsection will be devoted to the algorithm parameters and evaluation considerations.

### 4.1 The MCF idea

Let us consider one of the dedicated nodes $d \in D$ with, say, 4 adjacent nodes. Let us imagine

also that costs of outgoing links are subsequently $c(d, \ldots) = \langle 3, 4, 3, 7 \rangle$ (Fig.1).

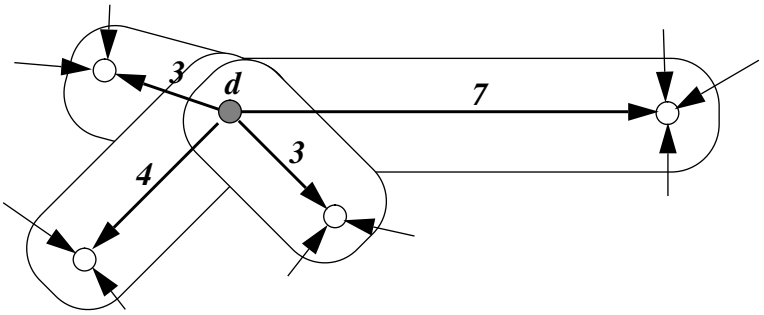


Fig. 1 The dedicated node connectivity levels

If we start selecting in the initial graph *G* all edges with minimum cost (equal to *1*); then with the next cost (equal to *2*) and up to the last cost value (equal to *K*) then in general case we'll obtain at each step a set of disconnected subgraphs. At step 3 the dedicated node *d* will also participate with 2 of it's edges in this subgraph, then at step 4 and at last at step 7 according to it's *cost pattern.* So will do all the dedicated nodes. We can't expect that any minimum cost spanning tree (and MCF as well) will connect the *d* using the *d*'s minimal cost links, however it seems reasonable to start checking this possibility with the lowest cost for each of the dedicated nodes.

It could be seen from Fig. 1 that starting with the dedicated nodes some connectivity levels will be utilized. At the same time we can observe what configurations could be produced starting with the rest of the network with the lowest cost. The MCF algorithm background arises from the idea of *simultaneous* movements to the desired minimum cost configuration from two directions. One direction (*TopDown*) is from dedicated nodes outwards the network; the other (*BottomUp*) - from all network nodes with minimum cost links through this link to enlarge minimum cost connectivity. According to the same cost values of links connecting different nodes in the network these nodes could be seen as participating in a sort of association (relation). The MCF tries to employ these associations for the sake of MCT construction. However this presentation needs additional formalization.

## 4.2 Formalization

### 4.2.1 Simplex Presentation

The MCT construction algorithm which is proposed in this paper is based almost totally on the network *simplex* presentation. All pairs of network nodes directly connected by links with communication cost equal to *k* are said to participate in a relation $\lambda_k(v_i,v_j)$. Such pair of nodes together with the communication link between them comprise the elementary building block (see Fig. 2 (a)) for the so called *simplex*. For the purposes of the algorithm design the simplex concept given in [30] is slightly modified. We call a simplex any connected subgraph $\sigma_i(k)$ of initial graph $G(V, E)$ with the relation $\lambda_k(v_i,v_j)$ between all adjacent nodes in $\sigma_i(k)$. Obviously any configuration of simplex (pair of adjacent nodes, chain, tree, mesh) can take place in $G(V, E)$. We call $\sigma_i(k)$ a *k-simplex*; index *i* could be omitted if this causes no confusion. Sometimes we make use of index *d* to emphasize that at least one of the d-nodes occurs in the simplex. In cases when we are not interested in simplex structure - but only in participating nodes - we list these nodes in angular brackets. And at last double indexes will be used for a simplex when we should like to make difference for simplexes with common node in each of

them.

The relation $\lambda_k(v_i,v_j)$ taken for the whole network results in the so called complex $X_k = \Lambda_k[G(V,E)]$ which is obviously a subgraph of $G(V, E)$ maybe partially connected. Any $i$-th connected part of $X_k$ is evidently a simplex $\sigma_i(k)$ so that $X_k = \bigcup_{i=1}^{ns(k)} \sigma_i(k)$, where $ns(k)$ - number of connected parts (simplexes) in $k$-complex. Strictly speaking about $k$-complex we should consider those nodes of initial graph which have no links with cost $k$ also as parts of $k$-complex. However we omit this detail here. Typical $k$-complex is presented in Figure 2 (b)

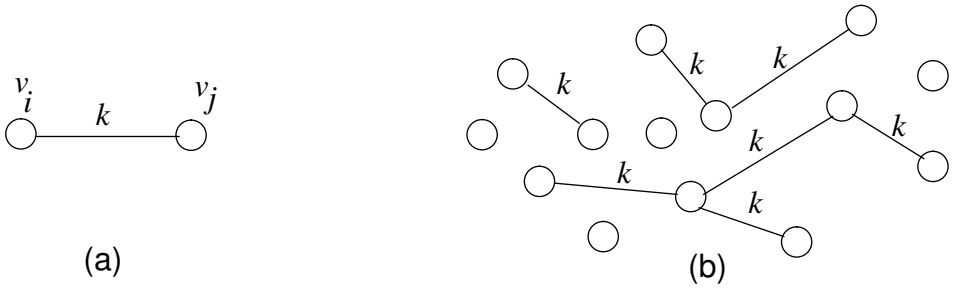


Fig. 2 Minimal simplex (a) and typical complex (b)

### 4.2.2 Stick Operation

Simplex gives not only a new name for an ordinary one-hop connectivity within a graph but also gives a new vision of the connectivity problem itself. According to the above definition there could not be common elements (neither vertexes nor edges) for any pair of $\sigma_i(k)$ and $\sigma_j(k)$ belonging to the same $X_k$. Otherwise they would comprise a single simplex. However there could be common vertexes (one or more) for some pair $\sigma_i(k)$ and $\sigma_j(p)$, belonging to different complexes $X_k$ and $X_p$ of the same graph $G(V, E)$. We shall say in this case that simplex $\sigma_j(p)$ *sticks into* simplex $\sigma_i(k)$ if $p > k$: $(\sigma_j(p) \bullet \sigma_i(k))$ and vice versa if $p < k$ (Fig. 3). Note that while the complex presentation "destroys" the initial connectivity of the graph the

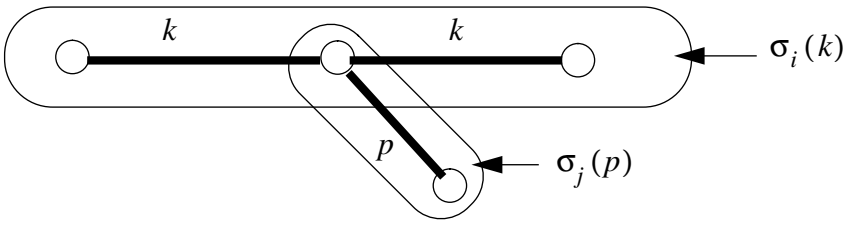


Fig. 3 Binary stick operation

*stick* operation improves connectivity using this new presentation of a network. The stick operation for any two simplexes may result in:

$$\sigma_j(p) \bullet \sigma_i(k) = \begin{cases} \{\sigma_i(k), \sigma_j(p)\} & \text{, if there are no common vertexes} \\ s\sigma_{i,j}(k,p) & \text{, if at least one common vertex exists} \end{cases} \quad (1)$$

If the stick operation is successful then the new object appears, the so called *sticked simplex*

$s\sigma_{i,j}(k,p)$ - a connected subgraph of $G(V, E)$ with any of two cost values $k$ and $p$ between any pair of adjacent vertexes. At least one path exists in any sticked simplex for any pair of vertexes belonging to any of simplexes participating in the stick operation.

Mentioned above example is limited to *binary* stick operation. We treat here binary twofold: first, only two simplexes participate in the stick operation shown above, and, second, only two cost values are considered. However in practice we can expect more successful stick cases resulting in chain, tree or any mesh of simplexes. Actually stick operation could be considered for complexes and treated as multiple stick operation $X_k \therefore X_p$, where

$$X_k \therefore X_p = \bigcup_{i,j=1}^{ns(k),\,ns(p)} (\sigma_i(k) \bullet \sigma_j(p)) \qquad (2)$$

i.e. a combination of results for stick operations applied to all possible pairs of simplexes taken from both complexes.

Figure 4 (a) shows an example of non-binary (in sense of number os simplexes) stick operation, while Figure 4 (b) shows more general example - non-binary in both senses: number of cost relations and number of simplexes are equal to 3.

We are reasoning in favour of MCT construction algorithm which will increase connectivity with minimal cost. That is why every time when a sequence of cost operations will be considered we'll assume that this sequence is ordered incrementally regarding the cost of it's operands. Furthermore, let us define $S_X(k, p) = X_k \therefore X_p$ as a result (subgraph) of stick operation applied to *any* 2 complexes and $S_X(k) = S_X(p) \therefore X_k | p = k-1$ as a result of *regular* stick operation, where $S_X(0) = \varnothing$, which is called in the algorithm ***BottomUp***:

$$S_X(1) = X_1 \qquad S_X(2) = S_X(S_X(1),2) \qquad \ldots \qquad S_X(K) = S_X(S_X(K-1),K) \qquad (3).$$

The result of regular stick operation becomes more and more alike the initial graph and obviously $S_X(K) = G(V, E)$. Similar to this we can define the result of *irregular* stick operation which will "accumulate" obtained connectivity when only a subset of simplexes, (say, those which contain d-nodes $\{\sigma_d(k)\}$ of $k$-th complex) will participate in the ***TopDown*** operation:

$$S_\sigma(1, p) = X_1 \therefore \{\sigma_d(p)\} \qquad \ldots \qquad S_\sigma(k, p) = X_k \therefore \{\sigma_d(p)\} \qquad (4) \quad .$$

The $\lambda_k(v_i,v_j)$ relation as well as stick result (sticked simplex) and multiple stick result (sticked

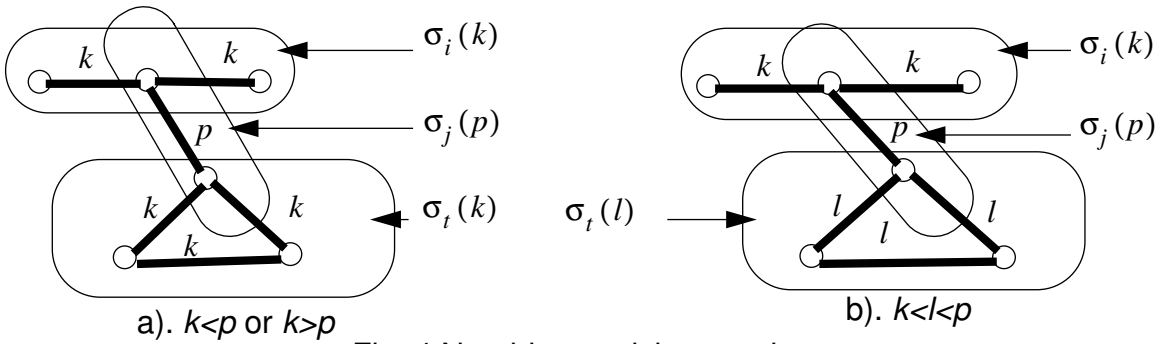


Fig. 4 Non-binary stick operation

complex) are particular presentations of a more general *connectivity* relation, i.e they all inherit all known properties of this relation given in a set of graph vertexes (network nodes); they are reflexive, symmetrical (for bidirectional graph) and transitive. Applying (1) to (2) and denoting the more general result of non-binary sticked complexes as $SX_{k_1, k_2, \ldots, k_T}$, where $T$ is the

number of sticked simplexes for the sequence of costs $k_1, k_2, \ldots, k_T$ we obtain (omitting non-sticked simplexes):

$$SX_{k_1, k_2, \ldots, k_T} = \bigcup s\sigma_{i_1, i_2, \ldots, i_{ns(T)}} (k_1, k_2, \ldots, k_T) .$$

### 4.2.3 d-Connectivity Metrics

Let us remember now that we have two kinds of vertexes in $G(V, E)$ , i. e. $V = D \bigcup (V \backslash D)$ and we are mostly interested in increasing of *dedicated connectivity* (d-*connectivity*) in $G(V, E)$ , that is connectivity of d-vertexes with minimal cost. Obviously any d-vertex belongs to a number of simplexes $d \in \{ \{\sigma_d(k_1)\}, \ldots, \{\sigma_d(k_{nc(d)})\} \}$ , where *nc(d)* is a number of cost values of links adjacent to a particular d-vertex. Let us denote $n(v_i)$ - *i-th* vertex' degree, the total number of edges adjacent to this vertex. Then *n(d)* will be equal to the total number of participation of *d* in different simplexes *iff* all cost values for it's links are different.

According to the d-connectivity definition we have to enlarge the connectivity of those simplexes which contain one or more d-vertexes in them. In Figure 5 we show a simplex

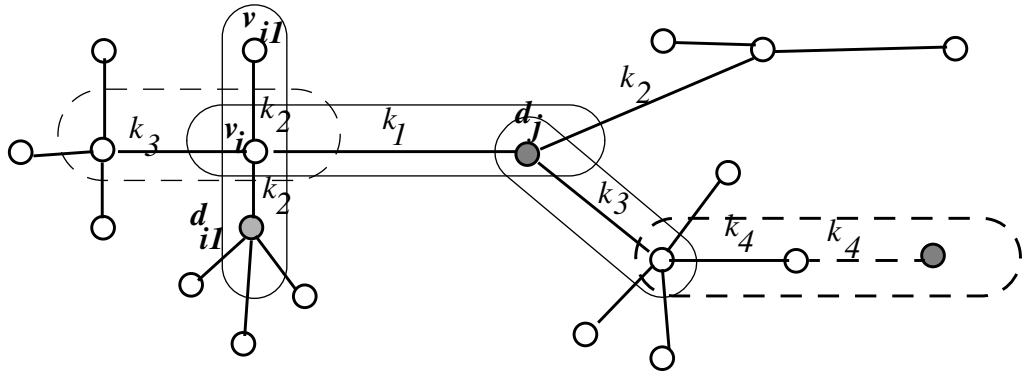


Fig. 5 One step d-connectivity increase

$\sigma(k_1) = \langle v_i, d_j \rangle$ which sticks into the simplex $\sigma(k_2) = \langle v_{i1}, v_i, d_{i1} \rangle$ . As could be seen from this example any other stick operation with initial $\sigma(k_1)$ doesn't enlarge it's d-connectivity in one step. In this example we assume that $k_1 < k_2 < k_3 < k_4$ and also that d-vertexes are represented with shadowed circles.

Let us call the following ratio as *eccentricity* of a *simplex*:

$$ecc(\sigma_d(k_1)) = \frac{k_2 - k_1}{K - k_1} \qquad (5)$$

where *K* is maximal cost for a given graph; $k_2$ - minimal cost increasing the d-connectivity of the given simplex; $k_1$ - cost of the given simplex. Simplex' eccentricity given by (5) shows the relative difference of simplex cost and that minimal cost of "adjacent" simplexes which enlarges the d-connectivity of the given simplex. We call "adjacent" these simplexes which could be sticked into the given simplex in one step, i.e. applying the stick operation once.

The d-*vertex eccentricity* (*ecc(d)*) will be the minimal eccentricity of all it's simplexes:

Under the assumptions that $k_1 \le k_2$ ; $k_1 < K$ and $k_1, k_2$ - integer; $k_1, k_2 > 0$ ; the following

$$ecc(d_j) = min\{ecc[\sigma_d(k_1)], \ldots, ecc[\sigma_d(k_{nc(d)})]\} \quad (6)$$

expression $1-ecc(\sigma_d(k_1))$ can be treated as a *probability* of the d-connectivity increase for the given simplex regarding only one step stick operation.

Eccentricity seems to be valuable metric and could be used instead of intuitively useful *density* of d-vertexes. For example, the density of d-vertexes could be very high for some network area which is totally isolated by high communication cost barrier from the rest of the network. Eccentricity shows exactly the relative position of a simplex or node within the whole network with regard to it's cost structure. In one particular case, when costs of all adjacent links of a d-node are equal (to $k_1$) we can conclude that expression (6) is equal to (5): in this case all values of $\{k_2\}$ are always equal to a single value (otherwise we should have different values in $\{k_1\}$ and consequently, several simplexes for the d-node).

Values $\{k_2\}$ and $\{k_1\}$ are unknown, however they could be estimated with the use of known costs for all d-nodes. Let us denote such estimations as $\hat{k}_1, \hat{k}_2$ and define as minimal and maximal cost of links adjacent to the given d-node. Under such estimations we roughly can use (5) instead of (6) which gives us simple formula to calculate the probability of d-connectivity increase in one stick step: $\hat{Pr}(d) = \frac{K - \hat{k}_2}{K - \hat{k}_1}$. If the maximum cost difference for links adjacent to a node will be called *node cost diameter* $(cdiam(d))$ and the difference between the maximal network cost and maximal node cost as node cost boundary $(cbound(d))$ then

$$\hat{Pr}(d) = (cbound(d)) / (cbound(d) - cdiam(d)) .$$

## 4.3 Cost Calculation for Shared MCT

In this section we show how the communication costs could be calculated for shared MCT construction algorithm taking into account that sources can send their flows simultaneously, the network is heterogeneous (in the sense of link's bandwidth) and required bandwidth could also be different for various senders.

When the MCT is constructed for the MC session with a single sender then the bandwidth needed (*b*) to guarantee the quality of this single flow is to be allocated at all edges of the MCT. In this case the MCT could be presented as unidirectional graph with weighted edges. Weights could be interpreted as costs of allocation of needed bandwidth. Let us note here that both bandwidth and delay interpretations of communication cost are direction dependant. That is why we always have to start with future sender when constructing a source-based MCT. On contrary, when constructing a shared MCT for highly dynamic (in the sense of floor management) collaborative applications we use to work with bidirectional graph. That is every edge could be a bidirectional link. Compare both presentation (see Figure 6 (a) and 6 (b)).

Each link of shared MCT shown in Fig. 6 (b) has to have allocated capacity needed for $N_d - 1$ flows, where $N_d$ is the number of dedicated nodes including current sender. Here we consider several senders transmitting simultaneously different flows. Let us assume that in the MCT shown in Fig. 6 (c) simultaneously exist 3 sources with various bandwidth requirements: d0 - *b0*; d2 - *b2* and d4 - *b4*. From Figure 6 (c) we can see that in this case all receivers, connected to shared MCT use the same bandwidth equal to *b0* + *b1* + *b4*.

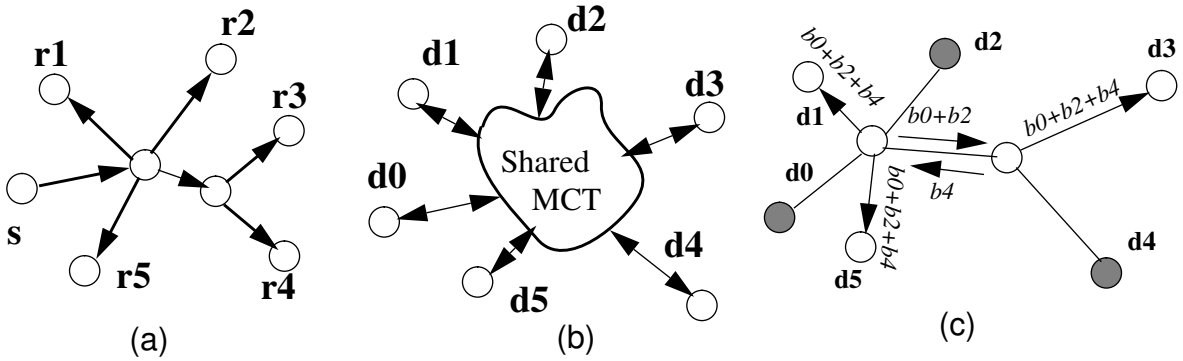


Fig. 6 Comparison of source-based (a.) and shared (b.) MCT

Let us make some assumptions which will be used for the rest of the paper if opposite will not be specially mentioned:

A1. The default bandwidth needed for each sender is *b*;

A2. Communication cost assigned to each link (edge in the graph) will be defined by the link bandwidth utilization, i.e. cost is set proportionally to available (free) bandwidth;

A3. Maximal acceptable cost equal to *K* cost units is assigned to a link with free bandwidth equal to *b*.

Let $\varepsilon$ denote the link cost increase within the established MCT due to a single flow allocation via the link, then for a shared MCT we'll have:

$$Cost(MCT_b) = Cost(MCT_0) + M_d \cdot n_s \cdot \varepsilon \qquad (7)$$

where $n_s$ - number os senders $n_s \le N_d$; $M_d$ - number of edges in the MCT, and $Cost(MCT_0) = \sum_{i=1}^{M_d} c_i$, where $c_i$ - initial value of the communication cost for *i*-th link.

Different MC sessions may require different default bandwidth *b*, that means in general that communication cost is to be recalculated after receiving the request with needed value according to following considerations.

Let us denote *B* - the total bandwidth of communication link (we omit here the index for particular link); *K* - number of cost levels for the whole network. At the same time, according to assumption A3 *K* is equal to the cost of the link with utilization:

$\frac{B-b}{B} = 1 - \frac{b}{B}$, that is $K = \rho \cdot \left(1 - \frac{b}{B}\right), \rho = \frac{K}{1 - \frac{b}{B}}$ where $\rho$ is some scaling coefficient.

Further for any link with cost $k<K$: $k = \rho \cdot LinkUtilization = \left(\rho \cdot \frac{B^c}{B}\right)$, where $B^c$ - currently available bandwidth for the link according to (7). The new cost of the link after allocation of MC default bandwidth *b* will be: $k^n = k + \varepsilon = \rho \cdot \frac{B^c + b}{B} = k + \rho - K$ and consequently $\varepsilon = \rho - K$. It is desirable for the calculation purposes to have $\varepsilon = 1$, that is the number of the cost levels which are to be assigned to all links could be counted as $K = \frac{B-b}{B}$.

However in the case of *different* bandwidths for network links we face a problem of cost levels alignment for the whole network. In this case we can suggest to work with the ordered list of

link's bandwidths: $B_0 \geq B_1 \geq \ldots \geq B_{M_T}$, where each $B_i = \alpha_i \cdot B_0; (\alpha_i < 1)$ ; $M_T$ - number of links observed during the MCT construction; and assign $K = \frac{B_0 - b}{b}$. For another (less) values of the link bandwidth we can suggest to consider the difference $B_0 - B_i$ as "always occupied" portion of the link bandwidth. To work with integers we also can suggest to take the nearest integer for any division.

Further if flow specifications for all senders in future MC session are equal to *b*, then we can use the link costs defined by needed bandwidth equal to $n_s \cdot b = b_c$.

If needed bandwidths differ for different senders of the MC flow then generally we'll have various cost levels; obviously this is undesirable. To have the same number of cost levels even in the case of different bandwidth requirements let us order these requirements: $b_0 < b_1 < \ldots < b_{n_s}$, i.e $b_i = \beta_i \cdot b_0$; $(\beta_i > 1)$ . The number of cost levels should be defined as $K = \frac{B_0 - b_0}{b_0}$ while each *i*-th flow still uses bandwidth $\beta_i \cdot b_0$ .

When constructing the MCT with known number of senders with different flow specifications for each sender we should use in the algorithm for the link cost calculation needed bandwidth as $b_c = \sum_{i=1}^{n_s} \beta_i \cdot b_0$.

## 4.4 The MCF algorithm

Using the notation explained earlier the MCF algorithm could be presented as follows:

```
k := 1; S_X(1) = X_1 ; stop := false; BU_Cost(k) :=1; i :=1; j :=1;        (*Init*)
```

```
while (k ≤ K) and (stop=false) do                                          (*Main*)
begin
    if (k + 1 ∈ DCB)
    then begin
        TopDown(k+1); TD_Cost(i) := k+1; i := i+1
    end
    else begin
        BottomUp(k+1); BU_Cost(j) := k+1; j := j+1
    end;
    k := k+1;
    G_f = { sσ_d | s(σ_d) ∈ [S_X(k) ∪ S_σ(k−1, k)] }   (* "Growing" MCT*)
    if (∀i = 1, N_d | d_i ∈ G_f) and (G_f - connected)
    then begin Prune(G_f); stop := true end;
end;
```

```
h := 1; S_X = G_f;                                                         (*Check*)
while (Cost(G_f) > MaxMCTCost) and (TD_Cost(h) < K) do
begin BottomUp[TD_Cost(h)]; h := h+1; end.
```

Comments: the value of MaxMCTCost is considered known; the dedicated cost base (DCB) is explained in the next session; TopDown is a regular stick operation defined in (3), while the BottomUp is irregular stick operation defined in (4); *Prune* is obvious operation of "tails" eliminating, while another kind of pruning - eliminating of alternative paths (loops) with larger costs - is made during the MCT growth.

## 4.4 Evaluation

The maximal number of operations for the Main part of the algorithm is:

$$K \cdot (N_d \cdot (n_{max}(d) + 1) + N) ,$$

where $n_{max}(d)$ - a maximal degree for dedicated nodes; while the number of operations for the Check part could be estimated only for some particular cases.
In order to find Check conditions for the MCF Algorithm let us examine the so called *cost patterns* of all d-nodes in a form of the following matrix:

$$\begin{bmatrix} \{k_{1,1}\} & \{k_{1,2}\} & \dots & \{k_{1,nc(1)}\} & \dots & 0 \\ \{k_{2,1}\} & \{k_{2,2}\} & \dots & \{k_{2,nc(2)}\} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \{k_{i,nc(i)}\} \\ \{k_{N_d,1}\} & \{k_{N_d,2}\} & \dots & \{k_{N_d,nc(N_d)}\} & \dots & 0 \end{bmatrix} \quad (8)$$

According to different numbers of various costs for various d-nodes (*nc(i)*) the number of non-empty elements will be different for every row. We assume that each matrix element is a "set" of equal cost values; all non-empty elements are left-adjusted and "leading" elements equal to 1 are omitted for their obvious participation in the BottomUp procedure which starts the MCF algorithm. Each i-th time when the TopDown procedure will run it will utilize one column of the matrix, i.e it will run non-regular stick operation over previous result and those simplexes containing d-nodes which correspond to a current cost for each d-node (*TD_Cost(i)*). Each of these sets of costs could be continuous or not, moreover they can overlap (Fig. C.1).

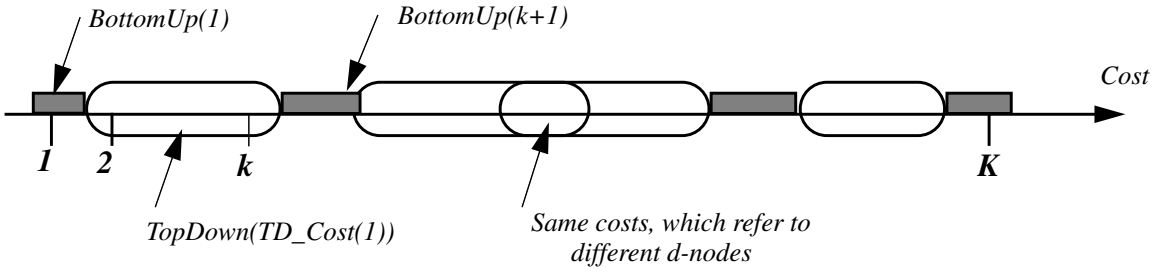


Fig. 7 Sample sequence of MCF steps

Particular sequence of steps for the MCF algorithm running over given network could be estimated with the use of particular relation between the network *cost base* (general sequence of all costs *1, ..., K*) and the so called *dedicated cost base* which considers only d-nodes.Figure 8 shows 6 typical cases (DCB1, ..., DCB6).

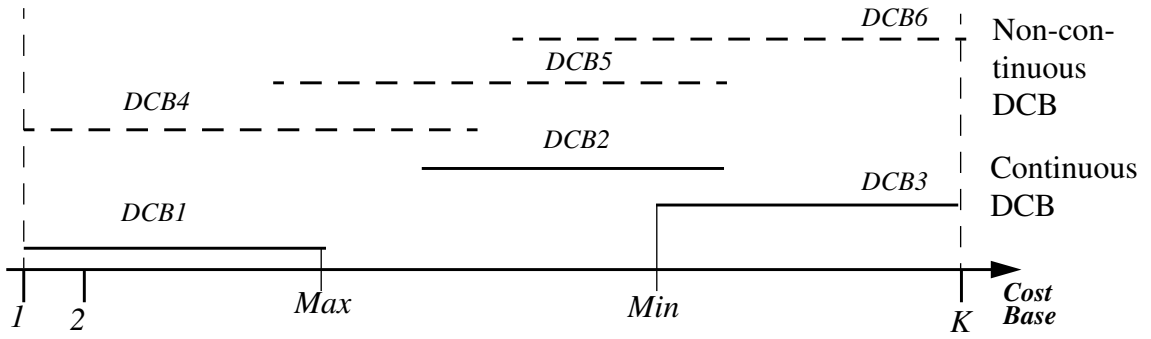


Fig. 8 Possible interrelations between CB and DCB

While constructing the MCF we can hardly expect to achieve needed tree consisting only of d-nodes. However intermediate simple nodes (s-nodes) are treated by the MCF algorithm also as dedicated nodes starting with the moment when this fact is realised by the algorithm. The number of these s-nodes will be denoted as *newd*. The particular value of *newd* characterises the length of paths between d-nodes (in a number of hops). Let us imagine that MCT is constructed. Whether it is optimal or not? In order to check this we can try those possibilities which were missed by the algorithm. According to the value of *newd* some check conditions could be formulated. We'll examine one arbitrary sticked d-simplex belonging to the MCT and shall try to find those cases when it's cost could be reduced in principle.

**Case 1.** If *newd=0* then this arbitrary d-simplex consists of two directly connected d-nodes:

$$\sigma_d(k) = \langle d1, d2 \rangle .$$

Let us suppose that our algorithm have missed better path tracing over some node *v*, i.e. (according to Fig. 9 (a)):

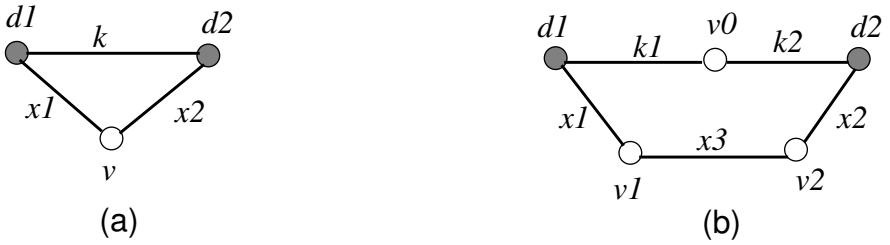


Fig. 9 Cost considerations for alternatives

$$x1 + x2 \le k - C_{CE} - C_{Sw},$$

where the right part of this inequality could be explained as follows. First, we assume to choose the path with minimal number of hops even with the same communication cost (here it is bandwidth allocation cost) for both paths in order to avoid the connection establishment cost $(C_{CE})$. Second, alternative path utilizes one more switch which causes additional switching cost $(C_{Sw})$. Following the considerations of [29] we set $C_{CE} = C_{Sw} = 1$. This gives:

$$x1 + x2 \le k - 2 .$$

That is in order to have better alternative for a d-simplex like *<d1,d2>* within the MCT we have to check BottomUp and TopDown steps of the algorithm (if any) with the following missed costs: *1, 2, ..., k-2*. Obviously these considerations are not needed really for such kinds of d-simplexes, because alternative path will be found by one of the TopDown steps of the algorithm. However the same reasoning will be applied for other cases.

**Case 2**. If *newd=1* then we have to seek for alternatives to a d-simplexes like

$$\sigma_d(k) = \langle d1, v0, d2 \rangle \text{ or } \sigma_d(k1) = \langle d1, v0 \rangle, \sigma_d(k2) = \langle d2, v0 \rangle \quad (9) .$$

Note that all alternatives like

$$\sigma_d(x) = \langle d1, v1, d2\rangle,\ (\sigma_d(x1) = \langle d1, v1\rangle, \sigma_d(x2) = \langle d2, v1\rangle)$$

will be found by the algorithm. In order for alternatives (9) to exist the following inequality is to be fulfilled (according to Fig. 9 (b) and variants of (9)):

$$\left\{\begin{matrix} x1 + x2 + x3 \\ 2 \cdot x1 + x2 \\ x1 + 2 \cdot x2 \\ 3 \cdot x \end{matrix}\right\} \le \{\begin{matrix} k1 + k2 \\ 2k \end{matrix}\} - 2.$$

For the most general case shown in Fig. 9 (b) we'll have

$$x1 + x2 + x3 \le k1 + k2 + 2.$$

Because we are looking for missed simplexes we may assume that *x3>1*, otherwise it should be found by the initial *BottomUp(1)* step. Let *x3=2* then

$$(x1 - k1) + (x2 - k2) \le 4 \qquad (10)$$

The check condition (10) means that we have to check only those pairs of d-nodes connected via 2 hops which have: i). Degree at least equal to 2; ii). Cost diameters sum more or equal to 4. These are rather strong conditions. For example, in the current MBONE structure [31] we can find hardly more than 5% of such nodes.

**Case 3.** Generally we may have any number of new dedicated nodes in MCT between a pair of d-nodes, say, *newd*. Then, according to Fig. 10 the check condition will be:

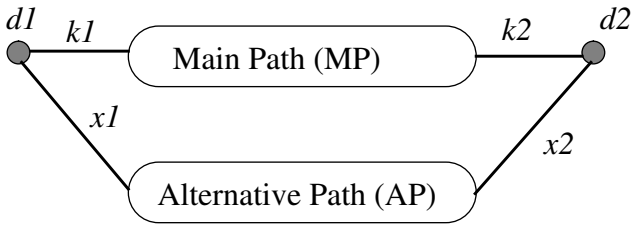


Fig. 10 General case for check condition

$$(x1 - k1) + (x2 - k2) \le Cost(AP) - Cost(MP) \qquad (11)$$

Repeating the above reasoning it easily could be interpreted that the check condition for (C.4) is as follows: *missing at least one BottomUp step in the algorithm*. However this most strict limitation of the algorithm in most practical cases could be "smoothed" by known cost structure (Fig. 8).

In general, efficiency of the Checking part of the algorithm could be estimated with the use of the ratio ($\rho$) of *sizes* of checking and total *problem spaces* over which the algorithm runs it's operations. The total problem space size is given by the number of nodes (*N*) and the number of costs (*K*), while the size of the checking space depends on possible interrelations of CB and DCB. For two cases it's easy to evaluate this ratio (look at Fig. C.2):

DCB1:
$$\rho = \frac{(K - Max) \cdot N + Max \cdot (N - N_d) + \sum_{i=1}^{newd} (Max - k_i)}{K \cdot N};$$

DCB3:
$$\rho = \frac{(K - Min) \cdot (N - N_d) + \left( \sum_{i=1}^{newd} (K - k_i) \right)}{K \cdot N}.$$

These formulae give for a network with 100 nodes, 20 dedicated nodes, 10 cost levels and *Max=Min=5* the checking overhead ratio for DCB1 equal to 0.9 and for DCB3 equal to 0.4.

This shows clearly how the algorithm is cost dependant.

## 5. Numerical Example.

To give an impression of how the MCF algorithm works let us apply it to the graph depicted in Figure 11.

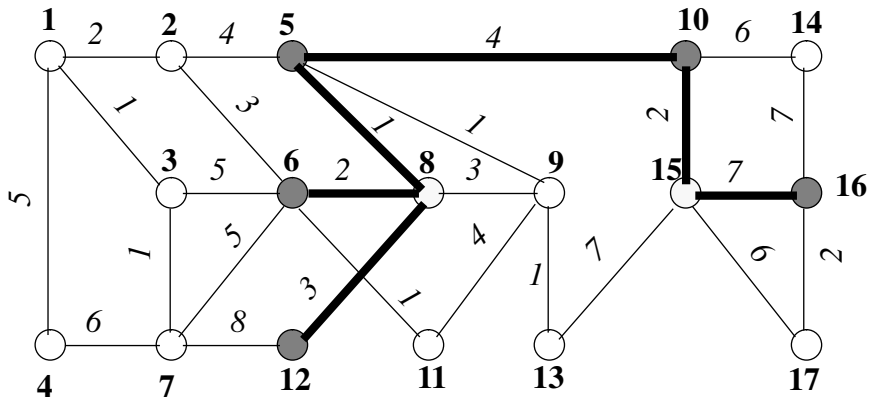


Fig. 11 Sample network graph

The resulting multicast connectivity is shown in bold edges of the graph. Initial graph presentation is given in Table 1 and resulting data structure depicted in Table 2.

**Table 1**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | | | 3<br>1 | 2<br>2 | | | | | | 4<br>5 | | | | | |
| **2** | | | | 1<br>2 | | 6<br>3 | | 5<br>4 | | | | | | | |
| **3** | | 1<br>1 | 7<br>1 | | | | | | | 6<br>5 | | | | | |
| **4** | | | | | | | | | | 1<br>5 | 7<br>6 | | | | |
| **5** | | 8<br>1 | 9<br>1 | | | | 2<br>4 | 10<br>4 | | | | | | | |
| **6** | | | 11<br>1 | 8<br>2 | | 2<br>3 | | | 3<br>5 | 7<br>5 | | | | | |
| **7** | | | 3<br>1 | | | | | | | 6<br>5 | 4<br>6 | | | | 12<br>8 |
| **8** | | | 5<br>1 | 6<br>2 | 9<br>3 | 12<br>3 | | | | | | | | | |
| **9** | 5<br>1 | | 13<br>1 | | 8<br>3 | | | 11<br>4 | | | | | | | |
| **10** | | | | 15<br>2 | | | | 5<br>4 | | | 14<br>6 | | | | |
| **11** | | | 6<br>1 | | | | | 9<br>4 | | | | | | | |
| **12** | | | | | | 8<br>3 | | | | | | | | | 7<br>8 |
| **13** | | | 9<br>1 | | | | | | | | | | | 15<br>7 | |
| **14** | | | | | | | | | | | 10<br>6 | | | 16<br>7 | |
| **15** | | | | 10<br>2 | | | | | | | | 13<br>7 | 16<br>7 | 17<br>7 | |
| **16** | | | | 17<br>2 | | | | | | | | | 14<br>7 | 15<br>7 | |
| **17** | | | | 16<br>2 | | | | | | | | | | 15<br>7 | |

Table 1 is a normal list presentation of a graph, however mentioning costs in the second line of each element and ordered according to cost increase; here every emphasized set of columns

presents a complex $X_k$ of the initial graph.

**Table 2**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | | | | 4 | | | | | |
| 2 | | | | | | 6 | | | | |
| 3 | 1,7 | | | | 6 | | | | | |
| 4 | | | | | 1 | | | | | 7 |
| 5 | 8,9 | | | 2,10 | | | | | | |
| 6 | 11 | 8 | 2 | | 3,7 | 2 | | | | |
| 7 | 3 | | | | 6 | | | | 12 | 4 |
| 8 | 5 | | | | | | | | | |
| 9 | 5,13 | | | | | | | | | |
| 10 | | 15 | | 5 | | | | | | 14 |
| 11 | 6 | | | | | | | | | |
| 12 | | | 8 | | | | | | 7 | |
| 13 | 9 | | | | | | | | | |
| 14 | | | | | | | | 16 | | 10 |
| 15 | | | | | | | | 16 | | |
| 16 | | 17 | | | | | | 14,15 | | |
| 17 | | | | | | | | | | |
| cost | 1 | 2 | 3 | 4 | 5 | 3 | 4 | 7 | 8 | 6 |
| Alg | BU | TD | TD | TD | BU | TD | TD | TD | TD | BU |

Table 2 shows all steps of the MCF algorithm. (look at the bottom line). Rows, corresponding to dedicated nodes are shadowed. While the BU (BottomUp) step simply transfers needed column from Table 1 to Table 2, the TD (TopDown) step operates with a subset of corresponding column from Table 1 according to the structure of DCB, which according to matrix (8) is as follows:

$$\begin{bmatrix} \{1,1\} & \{4,4\} & 0 & 0 & 0 \\ 1 & 2 & 3 & \{5,5\} & 0 \\ 2 & 4 & 6 & 0 & 0 \\ 3 & 8 & 0 & 0 & 0 \\ 2 & \{7,7\} & 0 & 0 & 0 \end{bmatrix}$$

## 6. Future research:

Future research areas cover a broad spectrum of implementation issues. It seems reasonable to elaborate this algorithm for a distributed implementation. The simplex presentation is essential for such purpose. If every router participating in a multicast tree will hold his current state in a form of all simplexes in which it participates then it can respond reasonably on every request concerning both MCT establishment and MCT run-time join. Such a scenario keeps multicast flow senders free from the request processing overload and thus increase scalability of the algorithm. Another possible direction is to employ the integral measures (eccentricity and d-connectivity increase probability) introduced in the paper to optimize the sequence of the algorithm steps and to provide performance analysis of the algorithm for particular network environment. Clearly the proposed algorithm could be beautified in many directions.

## Acknowledgements:

The author is very thankful to GMD FOKUS for perfect research environment and to 4 anonymous reviewers of the early version of this paper for their friendly and valuable comments.

## References:

1. Winter P.: Steiner Problem in Networks: A Survey. *Networks*, vol.17(1987), pp.129-167
2. R. Perlman: *Interconnections. Bridges and routers*. Addison-Wesley; 1992, 389 p.
3. Doar M., Leslie I. *How Bad is Naïve Multicast Routing ?* In Proc. INFOCOM'93, pp. 82-89.
4. Biersack E., Nonnenmacher J. *WAVE: A New Multicast Routing Algorithm for Static and Dynamic Multicast Groups*, In proceedings of the 6th International Workshop on Network and Operating System Support for Digital Audio and Video, April, 1995.
5. Deering S., Estrin D., Farinacci D., Jacobson V., Liu Ch.-G., Wei L. *An Architecture for Wide-Area Multicast Routing,* In Proceedings of SIGCOMM 94, London.
6. Bhagwat P., Mishra P.P., Tripathi S.K.: *Effect of Topology on Performance of Reliable Multicast Communication*. In: *IEEE INFOCOM'94,* pp. 602 - 610
7. Mitzel D.J., Estrin D., Shenker S., Zhang L. : *An architectural comparison of ST-II and RSVP*, In: *IEEE INFOCOM'94*, pp. 716 - 725
8. Effelsberg W., Mueller-Menrad E.: *Dynamic Join and Leave for Real-Time Multicast*; Tech. Rep. TR-93-056, International Computer Science Institute, Berkley, CA, Oct. 1993
9. Wei L., Estrin D. *A Comparison of Multicast Trees and Algorithms* (submitted to INFOCOM'94); at http://www.research.att.com/cgi-bin/bibsearch_html?Wei+Estrin
10. Schooler E. M. *The Impact of Scaling on a Multimedia Connection Architecture* at ftp://ftp.isi.edu/pub/hpcc-papers/mmc/scaling.ps
11. Szyperski C., Ventre G. *Efficient Multicasting for Interactive Multimedia Applications*, TR-93-017, International Computer Science Institute, Berkeley.
12. Onvural R.O. *Asynchronous Transfer Mode Networks: Performance Issues*. Artech House, 260 p.
13. Waxman B.M. *Models for Multipoint Connections in Gigabit Networks*, In electronic Proceedings of the GBN'95 Workshop: http://info.gte.com/ieee-tcgn/conference/gbn95/waxman.abstract.txt, April, 1995.
14. Salama H.F., Viniotis I. , Reeves D.S., Sheu T.-L. *Multicast Routing Algorithms For High Speed Networks*, at ftp://ftp.csc.ncsu.edu/pub/rtcomm/RTMulticastSurvey.ps
15. Shukla S.B., Boyer E.B., Klinker J.E. *Multicast Tree Construction in Network Topologies with Asymmetric Link Loads*, Technical report NPS-EC-94-012, ftp://ftp.nps.navy.mil/pub/ece/shukla/nps-mltcst-asym-linksv1.ps
16. *MBONE Global Agenda* at http://www.cilea.it/MBone/agenda.html and at http://www.eit.com/techinfo/mbone/mbone.html (V. Kumar)
17. Kou L., Markowsky G., Berman L., *A Fast Algorithm for Steiner Trees*. Acta Informatics, Vol. 15: 141 - 145, 1981
18. Plesnik J. *Heuristics for the Steiner problem in graphs*. Discrete Applied Mathematics, 37/38 (1992): 451 - 461, North-Holland.
19. Deering S. *Host Extensions for IP Multicast*, RFC 1112, August,1989.
20. Deering S. (Ed.) *IGMP Router Discovery Messages*, RFC 1256, September, 1991
21. Deering S., Estrin D.,Farinacci D., Jakobsen V. *IGMP Router Esxtensions for Routing to Sparse Multicast-Groups*, Internet Draft, available e.g. at http://karin.elch.lu.se/pub/stan ... ft-ietf-idmr-igmp-sparce-00.txt
22. Moy J. *Multicast extensions to OSPF*, RFC 1584

23. Marlow D. *Host Group Extensions for CLNP Multicasting*, RFC 1768, March,1995.
24. *Support for Multicast over UNI 3.1 based ATM Networks.* Internet draft: draft-ietf-ipatm-ipmc-04.txt.
25. Cheung S. Y., Kumar A.: *Efficient Quorumcast Routing Algorithms*, In *IEEE INFOCOM'94*, pp. 840 - 842
26. Schacham N., Meditch J.S. *An Algorithm for Optimal Multicast of Multimedia Streams*, In IEEE INFOCOM'94, pp. 856 - 864
27. Ammar M.H. *Probabilistic Multicast: Generalizing the Multicast Paradigm to Improve Scalability.* In: *IEEE INFOCOM'94*, pp. 848 - 855.
28. Handley M., Wakeman I. *CCCP: Conference Control Channel Protocol. A scalable base for building conference control applications*. v.1.4. An html link was provided at http://www.research.att.com/cgi-bin/bibsearch_html?CCCP
29. Ammar M.H., Cheung S.Y., Scoglio C.M.: *Routing Multipoint Connections Using Virtual Paths in an ATM Network,* In: *IEEE INFOCOM'93*, pp. 98 - 105
30. Casti J. *Connectivity, Complexity and Catastrophe in Large-Scale Systems*, Wiley, 1979.
31. Casner S. *The MBONE Frequently Asked Questions,* at: http://www.research.att.com/mbone_faq.html