Efficient Multicast Routing in High Speed Networks

M. I. Smirnov[1]
Saint - Petersburg State Electrotechnical University
5 Prof. Popov str., St.Petersburg,
197376 Russia

Abstract

The paper presents jointly two algorithms: for a minimal cost shared multicast tree construction (MCTC) and for a scalable multicast session call establishment, called multicast tree development (MCTD). Both algorithms are of original design based on a network simplex presentation, providing new effective approach to the MCTC, and object-oriented modelling framework providing highest scalability for sparse groups with high dynamics of membership, different quality of service requirements and multiple senders. A new approach is proposed for the MCT construction: bidirectional growing of the MCT without pruning; instead the MCTD makes use of extra branches to improve reliability and advertise new receivers.
Key words: high-speed network routing; multimedia streams; multicast tree; object-oriented framework; call model; scalability.

1. Introduction

New applications supporting multimedia multipoint-to-multipoint collaboration are on their way to customers in various domains such as distributed production management, scientific collaboration, medicine, distributed computer aided design, and so forth. 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 and usually involves construction of a special *multicast [routing] tree (MCT)* with data packets being replicated only at splitting nodes of the MCT.

Flexibility, scalability and manageability of this support provided from the network will most probably become the key features for the applications success. That is the multicasting will become one of the essential network services and will most likely support the quality of service negotiation at the call establishment phase and quality of service assurance for the session lifetime; it will support high dynamics of the members group; it will support multiple senders option for various communication modes.

These desirable features will affect not only one particular issue of the multicasting but the whole multicasting system. This paper tends to address this system view along with the proposal of two algorithms - the effective multicast tree construction and the MCT development within the proposed object-oriented framework for the multicast session call establishment. Both of the algorithms were elaborated with a very few concepts which are essential to the multicast sessions. These ideas are: i). The multicast is about global connectivity, i.e all participants have to “move” towards one another (as people do gathering in a meeting room), so it’s reasonable to construct the MCT from leaves and root simultaneously; ii). Usually the MCT construction algorithms provide an exhaustive structure which needs to be pruned; the paper proposes to make use of this exhaustiveness (for reliability reasons and to avoid traffic concentration along shared MCT); iii). Important assumption: the network should be able to transfer the multicast software agents throughout the network, which is on it’s way [1, 39, 42].

The multicast agent is an object with a number of interfaces facilitating self delivery and

1. This study was undertaken while Dr. Smirnov was at GMD FOKUS (Berlin, Germany).

self installation; quality of service negotiation; multicast session control, etc., it is a product of a contract between the multicast session initiator (initial sender) and the network service provider. The paper considers only network routers participating in the MCT construction and MCT development.

The paper is organized as follows. Section 2 introduces some related work in areas of the MCT construction algorithms and call establishment models. Section 3 is devoted to the original MCTC algorithm (model, idea, formal background and evaluation), while section 4 presents the MCTD algorithm (idea, scenario, components, functionality specification). Appendix gives an example of joint use of the two algorithms. The paper is concluded with implementation remarks and needed future research areas.

2. Related Work

2.1 The MCT Construction

The multicast tree problem was intensively studied in the last decade from various viewpoints, including the MCT topology design [2 - 8] (going to an NP-complete Steiner problem in graphs), protocol issues for different network technologies [9 - 17], hardware design and special application problems [18 - 21] including the Internet working prototype (MBONE). In this paper we address the first issue having in mind, however the particular application communication pattern. 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 [2, 4, 5, 7, 8] about two dozens (including variants) of different heuristic algorithms could be found with the KMB [7] 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. Any paper presenting a new heuristic decision is expected at least to make a comparison with the KMB algorithm.

This paper follows 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 3.6. Another advantage is that no pruning is needed after receiving the solution: auxiliary low cost structure is used during the MCT development. 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 [8] 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.

The communication cost definition will influence greatly the final MCT, however the cost assignment problem has no unique decision [22, 23]. 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.

2.2 Call Establishment

The Multimedia Call Establishment [24] (a "global view" model) deals with several objects, *medium* being among the object set, including also *relation* (for synchronization purposes between different information streams) and *attachment* (to relate a user to a certain medium). This approach could not be overestimated because of the potential it gives to designers. However, after introducing these objects the authors deal only with service primitives and provide the explanation of their specification as if the Call objects were not introduced.

From the signalling viewpoint, one can consider trends in a call control area, such as separation of the Call Control from a Connection Control; introducing some building blocks in the Call Control information structure and generic signalling flow mechanism based on atomic actions [25] (local view approach). When the set of 7 elementary objects is introduced they are to be "combined and organized into composite structures that logically represent a service"[25, p.302].

In [26] the call model is considered to have 2 parts: Session Control State Model (SCSM) and the Connection Coordination Model. The SCSM provides control functions for one session (equal to one call) which actually can invoke more than one connection. The SCSM model consists of 11 Points In Call; where PIC is distinguished state in the state transition diagram indicating the flow of operations. The CCM describes the connection topology and types of connections in some abstract way in order to be network independent. This description is based actually on 4 objects. In [27], which is actually based on the same concepts and testbed as [25] the object oriented Call model is represented with the number of 12 objects grouped into 3 generic classes, namely Local Object (8 elementary objects), Shared Object (2 elementary objects) and Virtual Object (2 elementary objects). Notably that the number of messages is rather small (Request, Accept, Ack and Reject).

One can try also to derive the object model out of the concrete multicast protocol description for the so called Protocol Independent Multicast - PIM [9] (4 objects and 6 primitives).Another multicast implementation, Reliable Process Group Communication, deals with the group view maintained at each host (somewhat similar to a global view) and with a sort of multicast address hierarchy enabling group dynamics within a certain domain [29]. All group views are treated consistently due to the implementation of a "two-phase commit" scheme. In [30] the author suggests the new metaphor of the network providing a sort of a universal service for the multimedia objects; which services are also able to provide the multicast service. In [21] authors describe a scalable base for building conference control applications and list number of urgently needed primitives of the multicast.

Sometimes it is hard to define exactly the set of call primitives used in different approaches; let us agree that call establishment capabilities will refer to generalized primitives in each model. It is evident from this short survey that there is no approach available for object definition in general for this kind of model, especially for the multicast case.

What the multicast system designer would like to have is, according to the author's opinion, some framework which will provide typical objects; their interrelations and some "method template" for every object. An attempt to provide such a framework for a multicast call establishment modelling was undertaken in this paper.

And, finally, the idea of network agents, travelling within the network and performing their tasks is not new, the latest complete overview of trends in the area could be found in [39], somewhat similar idea (proxy server) was implemented for the WWW service [40], while somewhat opposite task - migrating of user environment (teleportation) - was elaborated in [41]. Finally, it should be mentioned that a software migration itself seems to become one of the emerging areas of the software industry [1].

3. The MCT Construction Algorithm

3.1 Network 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 don'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.

3.2 The Algorithm Idea

Dissimilar criteria could be applied to the MCT, two major are delay and cost associated with the network links [22]. This paper deals only with the 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 [32, 33]. 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 [22]), 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 [9]. 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 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.

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 MCTC tries to employ these associations The MCTC is based almost totally on the network *simplex* presentation. Needed formal background could be found in Appendix 1. Informally we consider a network graph to be partitioned into sets of interconnected subgraphs, simplexes $\sigma_i(k)$, where *k* denotes the value of common communication cost for all links within the simplex, i - simplex's number. A set of all simplexes for the graph is called complex. If some simplexes have common elements (vertexes) we say that these simplexes are sticked. 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.

3.3 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 (look for Fig. 1 (a)) for the so called *simplex*. For the purposes of the algorithm design the simplex concept given in [38] 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.

A typical k-complex is presented in Figure 1 (b).

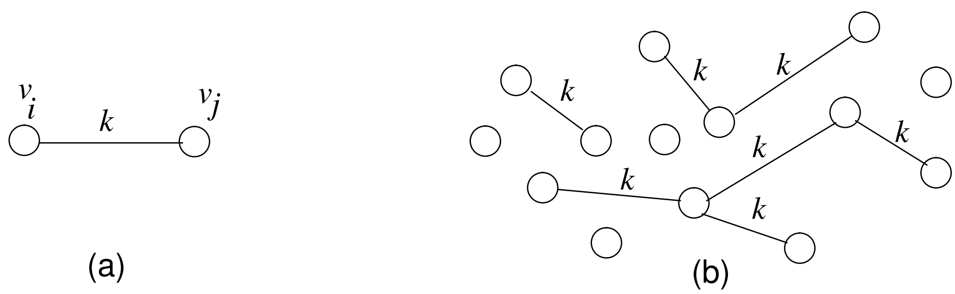


Fig. 1 Minimal simplex (a) and typical complex (b)

3.4 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 be no 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. 2). Note that while the complex presentation "destroys" the initial connectivity of the graph the

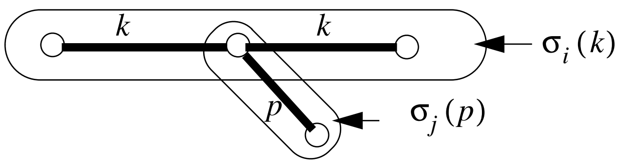


Fig. 2 Binary stick operation

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 a *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

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

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

Figure 3 (a) shows an example of a non-binary (in sense of number os simplexes) stick operation, while Figure 3 (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 a 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 operation is called in the algorithm ***BottomUp***:

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

The result of a 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 $\{\sigma_d(k)\}$ which contain d-nodes of the k-th complex) will participate in the ***TopDown*** operation:

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

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

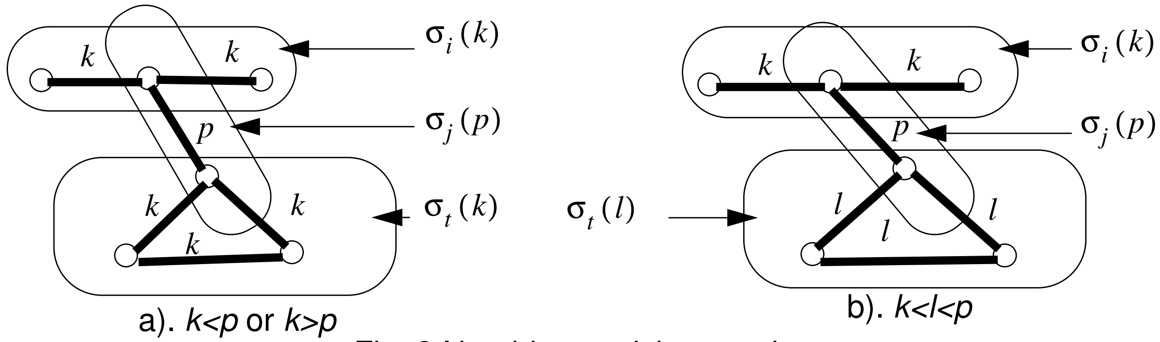


Fig. 3 Non-binary stick operation

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

3.5 d-Connectivity Metrics

We have two kinds of vertexes in $G(V, E)$, i. e. $V = D \cup (V \backslash D)$ 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 4 we show a simplex

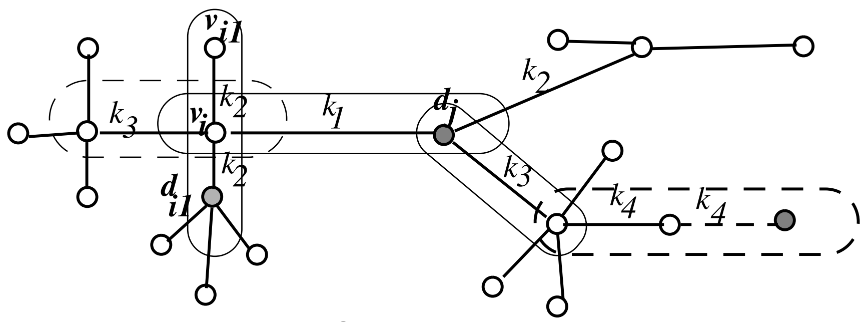


Fig. 4 One step d-connectivity increase

$\sigma(k_1) = \langle v_i, d_j \rangle$ 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 ratio (1) as *eccentricity* of a *simplex*: where *K* is maximal cost for a given

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

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:

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

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 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 a 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. Values $\{k_2\}$ and $\{k_1\}$ are unknown, however they could be estimated (as $\hat{k}_1, \hat{k}_2$) 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 a simple formula to calculate the probability of the d-connectivity increase in one stick step: $\hat{Pr}(d) = \dfrac{K - \hat{k}_2}{K - \hat{k}_1}$. If the maximum cost difference for links adjacent to a node will be called the *node cost diameter* $(cdiam(d))$ and the difference between the maximal network cost and maximal node cost as the *node cost boundary* $(cbound(d))$ then

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

3.6 Cost Calculation

We need this section to show how the algorithm can operate under various conditions with the same unique number of costs, represented by integers *1, ..., K*. 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; communication cost is defined by the network quasi-statically;

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

Detailed explanation could be found in [34] while here we summarise resulting expressions for the cost level calculation (Table 1).

Table 1:

Case description	Number of cost levels	Comment
Maximum N_d senders with needed bandwidth *b* for each sender	$K = \frac{B-b}{B}$	B - equal bandwidth for all links
N_d equal senders, heterogeneous links	$K = \frac{B_0-b}{b}$	$B_0 \geq B_1 \geq \ldots \geq B_{M_T}$, where M_T - number of links observed during the MCT construction
Heterogeneous senders and heterogeneous links	$K = \frac{B_0-b_0}{b_0}$	$b_0 < b_1 < \ldots < b_{n_s}$

3.7 The MCT Construction Algorithm (MCTC)

The MCTC algorithm is presented in Figure 5., where the following notation is used. *CP(i)*

```
k:=1;
while (k<=K) do
begin
  (* BottomUp *)
   D2(*,k):=D1(*,k);
   D1(*,k).used := true;
   CostCheck(k);
   k:=k+1;
  (* TopDown *)
   for i := 1 to N_d do
   begin
    k1:=1;
    if D1(i,k1).used=true and
    D1(i,k1+1).used=false
    then begin
          D2(i,k1+1):=D1(i,k1+1;
          D1(i,k1+1).used:=true;
          if k<k1+1 then k:=k1+1;
          CostCheck(k);
         end;
    end;
end.
```

```
CostCheck:;
for i:=1 to N_d do
for j:=i+1 to N_d do
   if {(i,j) are connected in D2} and
   {current_connect_cost < COST(i,j)}
   then begin
        COST(i,j).src(i).best:=false;
        COST(i,j).src(j).best:=false;
        D2(i,k).best:=true;
        D2(j,k).best:=true;
        COST(i,j):=current_connect_cost
       end;
```

Fig. 5The MCTC Algorithm

- the so called cost pattern of the $v_i \in V$, that is an array of costs for all outgoing links of the v_i in ascending order; $D1_{N \times K} = [d1(i,k)]$, $(i = \overline{1,N})$, $(k = \overline{1,K})$, where $d1(i,k)$ is a data structure $list(i,k)$ if $k \in CP(i)$ and *0* otherwise. The data structure $list(i,k)$ is an array of triplets $(v_i, k, used)$, where $\forall v_i \in V$; $e(v_i,v_j) \in E$ and $c(v_i,v_j) = k$. The boolean variable *used* is initially set to *false* for every element of the data structure. Obviously,

$$CP(i) = [list(i,k_1), list(i,k_2) \ldots, list(i,k_{nc(i)})]$$

D1 is a block matrix, where all rows corresponding to dedicated nodes are combined in the upper part for the sake of visibility (look for an example in Appendix). The D2 is the resulting matrix of the algorithm and has the same structure, except the boolean variable *used*, which is substituted by the boolean variable *best*, initially set to *false* for each element of D2.

The procedure CostCheck makes use of the square matrix $COST_{N_d \times N_d}$, which stores paths' costs between all pairs of d-nodes together with pointers to the beginnings of corresponding paths in D2 (*COST(i,j).src*); all cost elements of the matrix are set initially to infinity, all pointers are set initially to nil.

3.8 Evaluation

The algorithm computational complexity is: $O\left(\beta \cdot K \cdot \left(N_d^2 + N\right)\right)$, where β is a coefficient, which actually has a functional relation to the d-connectivity probability distribution across the set of d-nodes. The number of operations for the Check part could be estimated only for some particular cases. The Check part is not shown here (it's merely a repetition of the *BottomUp* procedure for the missed cost values).

Let us examine the so called *cost patterns* of all d-nodes in a form of the following matrix:

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

According to different numbers of different costs for various d-nodes (*nc(i)*) the number of non-empty elements will change 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 MCT 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. 6).

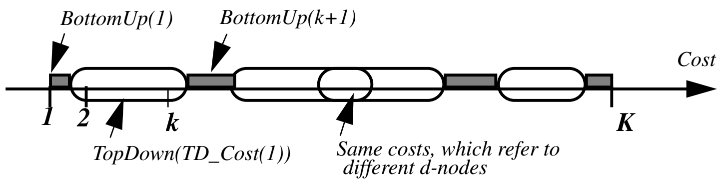


Fig. 6 Sample sequence of MCTC steps

Figure 7 shows 6 typical cases (DCB1, ..., DCB6) of relation between the network *cost base*

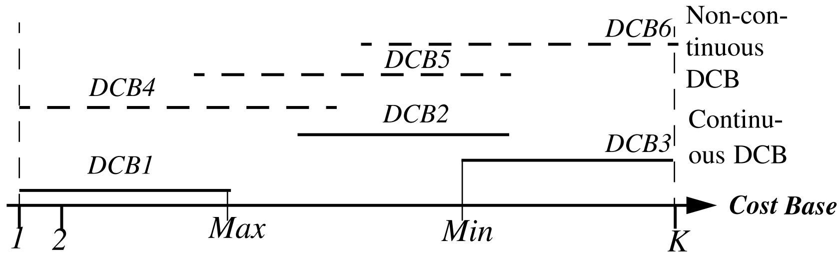


Fig. 7 Possible interrelations between CB and DCB

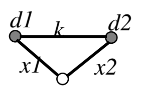
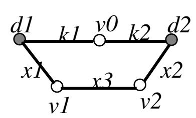
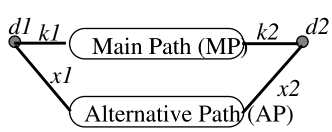
(general sequence of all costs *1, ..., K*) and the so called *dedicated cost base* which considers only d-nodes.

While constructing the MCT we can hardly expect to achieve needed tree consisting only of d-nodes. However intermediate simple nodes (s-nodes) are treated by the MCTC 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).

Whether MCT 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. We assume to choose the path with minimal number of hops even with the same communication cost (here it is the 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 [23] we set $C_{CE} = C_{Sw} = 1$. The checking conditions for the algorithm efficiency are presented in Table 2.

Repeating the above reasoning it easily could be interpreted that the check condition for the last case 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. 7).

Table 2:

newd	Case description	Check condition
0		$x1 + x2 \le k - C_{CE} - C_{Sw}$, and, under assumption: $x1 + x2 \le k - 2$
1		$x1 + x2 + x3 \le k1 + k2 + 2$; it's a strong condition. For example, in the current MBONE structure [35] we can find hardly more than 5% of such nodes.
newd		$(x1 - k1) + (x2 - k2) \le Cost(AP) - Cost(MP)$

In general, efficiency of the Checking part of the algorithm could be estimated with the use of the ratio (ρ) 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 (DCB1 and DCB3) it's easy to evaluate this ratio (look at Fig.7):

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

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

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.

4. The Multicast Tree Development

4.1 Idea and Background

The call establishment in the [multi]point-to-multipoint case is more complex than in point-to-point one. Introducing logical channels between the sender and each of the receivers of the multicast flow is probably a bad idea, especially for the call model because it hides essential features of the multicasting such as partial sharing of the multicast tree between receivers as well as the receiver group dynamic properties. The latter means that we can't separate in time the call establishment phase from the information transfer phase within the multicast, if receivers are allowed to join and leave the session at any time.

The proposed approach for the call modelling framework tries to separate logically entities participating in the call process as much as possible and specify their interfaces as well as object' methods behind them in such a way that real design could benefit from this separation in terms of model flexibility, completeness, feasibility and, mainly, manageability. The rationale behind this separation was perfectly introduced in [25]; it was stated there that such services as communication among heterogeneous terminal equipment; selectable quality of service; support of multiple connection calls; support of multiple party connections; individual control of mapping and presentation of information on each user's access, "cannot be modified effectively by a monolithic call object or single state machine. It would be necessary to enumerate a very long list of service and capability variations and the resulting finite state machine would be so enormous that it would be unmanageable" [25, p.302]. What probably new was introduced here is an attempt to derive necessary objects from the network support for multicasting.

4.2 The Multicast Case

The paper considers the multicast session as a planned event with one sender and a certain number of receivers who agreed to participate in advance (dedicated receivers, DR). The multicast session has certain parameters (DR list, QoS defaults) requiring certain services from the network, i.e. from the network service provider (SP).More formally the contract could be seen as having the following format:

```
multicast_session_contract(management_domain,
  initiator, QoS_requirements,
  [number_of_simultaneous_senders],
```

```
[session_duration], MCT, [modality],
[MCT_development_policy].
```

The management domain is needed within the contract because the service provider takes responsibility to assure the quality of service along the whole MCT and during the session's lifetime; the initiator obviously is to be mentioned within a contract, because at the contract establishment phase it is seen as a root of the MCT, however the MCT need not to be obligatory initiator-specific; the QoS could be specified in terms of adopted MCT quality criterion (cost or delay). The *MCT* parameter gives the network routing policy to reach all participants in the most effective way. The rest of the parameters are optional and will be explained later.

The SP has to confirm or to reject the requirements. Negotiation mechanism on service parameters with the SP is not considered here. When confirmed in a form of some contract the multicast call establishment request has to follow the multicast tree for the particular session, dedicated receivers being the leaves of this tree. If QoS parameters for some receivers differ from defaults then some router has to negotiate about new settings for these receivers. Some decision making is needed on when to start the session, how to prune branches, etc. However, if all this exchange of messages - patching of losses of packets, QoS negotiation, receiver group dynamics, floor and application control, etc. - will go throughout the whole MCT then this scheme will not scale well. Evidently, sooner or later the source node will become a bottleneck.

This paper proposes a highly scalable decision: i). Only one member of the multicast group negotiates initially with the SP, namely the so called initial source (initiator); ii). Receivers send their *acknowledgement, join* and *leave* messages to the nearest splitting point of the multicast flow; iii). All active senders share the same multicast tree (that's why the SP has to limit probably the maximal number of active sources); iv). All QoS negotiations with the receivers could be provided at the splitting points.

The call establishment thus continues during the information transfer phase of the session: notifications are still coming to the initial sender and to splitting points; advertised receivers start joining the session; some of them can decide to leave the session; others can decide to request their own flow specifications different from the default flow specification. We also assume that any of the dedicated receivers can become a sender to the same session. This option could be enabled by a special *floor request* message sent to the group. The only difference of the floor request from the initial sender request is that it doesn't need to be confirmed by the service provider. For some period of time we assume that there could be several senders for the same multicast session sending in parallel.

The multicast session is considered to enable connectivity of dedicated nodes where reside critical to the collaboration group members, however it could be the case that other participants may wish and be allowed to participate as well. This is a subject of the contract, however it seems helpful to consider 4 cases presented in the Table 3, assuming that any of the receivers

Table 3:

Send to World	Nearest to the initiator *ready to participate* nodes	The resulting configuration is unpredictable
Send to D-nodes	The result will depend on contract modality, availability of best routers and existence of alternatives	Depending on timeouts and alternatives the needed MCT will be established, however with high potential to growth
	Limited Contract	**Unlimited Contract**

linked currently to the MCT could potentially become a sender of it's own flow. Table 3 gives an overview of possible results of the MCT development. We consider that the multicast session can start only after receiving of acknowledgements from all of the dedicated nodes (if they are mentioned in a contract) or after expiry of the hierarchy of all of the timeouts (section 4.6). The multicast contract is called *limited* if the maximal number of receivers and/or simultaneous senders is limited; otherwise the contract is called *unlimited*.

It is important to distinguish between two kinds of routers participating in this MCT development scenario: best and advertised routers. Each of them will receive a copy of the MCA, however with slightly different functionalities. The difference is in the *modality* of their participation within the MCT development. While the best routers *must* participate and must develop their fanouts, the advertised routers *may* participate and may develop the MCT further, based on their local decision making process. If during the MCT development some fanout will experience the timeout expiry for it's best router, then it means that the MCT could not be developed further if this best router is the MCT splitting point. However, the fanout development algorithm presented in section 4.6 gives a solution for this problem which actually changes the splitting point based on any available alternative path which was not pruned during the MCT construction algorithm. Figure 8 explains this case. Note, however, that connection from the new split-

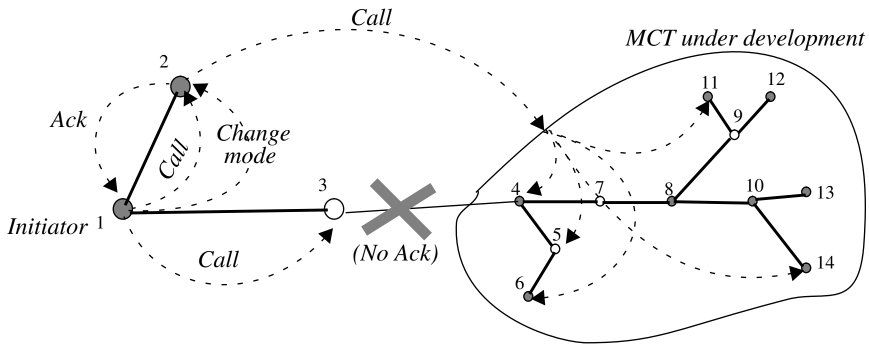


Fig. 8 Changing a splitting point in the MCT

ting point could be established to any router mentioned in the contract as a d-node or as an s-node (network nodes which belongs to the unicasts of the constructed MCT, next section).

4.3 MCT Decomposition

Any multicast tree could be seen as a composition of several serial interconnections of nodes (unicasts) and fanouts; that is any communication system supporting and providing multicast could be composed out of several unicasts and simple multicasts. The multicast is called simple when the corresponding multicast tree is a fanout. From these reasonings comes the idea of a scalable multicast. The multicast tree decomposition is presented in Figure 9, where simple composition components are put within smaller icons. Out of this example it is evident that due to the inevitable overlapping of unicasts and fanouts the functionalists needed to support unicasts and simple multicasts are to be placed within their intersection points, while these points are actually the splitting points of the multicast flow.

The desired scalability feature could be achieved via limited visibility out of each splitting point: we say that from each splitting point only a set of nearest reachable splitting points and receivers could be observed. This improves scalability and manageability of these partial multicasts. The need to provide the self-distribution of the multicast functionality over the multicast tree is the cost for these improvements.

In this paper we are especially aware of the case of the so called sparse group multicast [14], when the number of receivers is very large comparatively to the number of possible senders; receivers could also be distributed across several domains. In this case the multicast tree will probably have the following types of network nodes in it (Fig. 9, note that receiving hosts attached to each of the receiver nodes are not shown): initial source node (1); intermediate node without receivers attached to it (3, 5, 7); intermediate node with receivers attached to it (9); receiver node with receivers attached to it (4, 8, 10); leaf receiver node (2, 6, 11, 12, 13, 14).

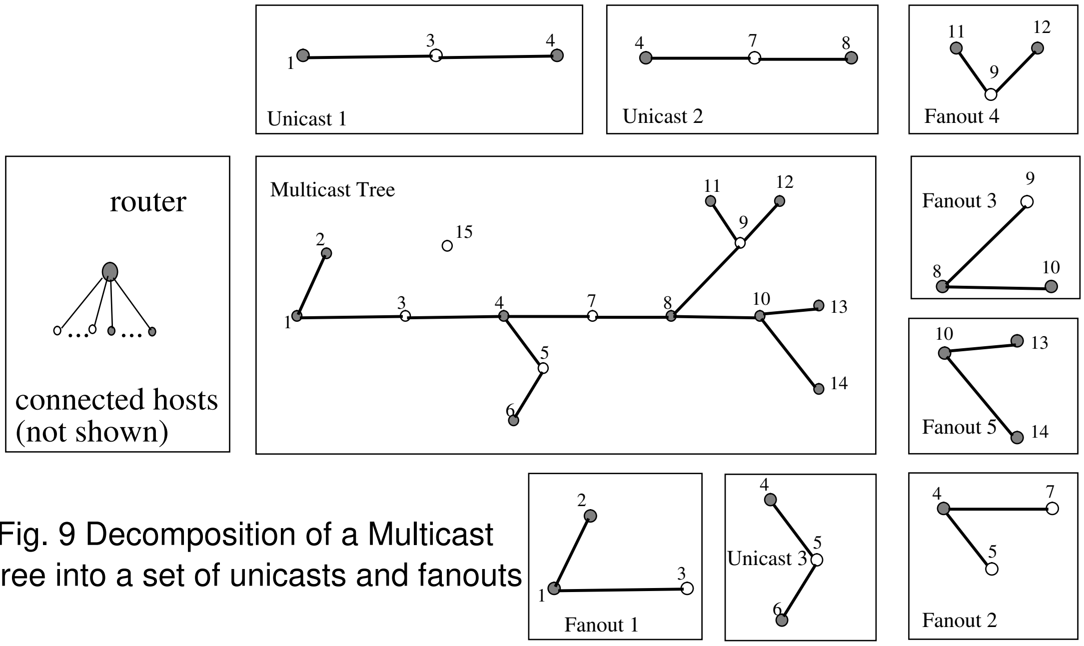


Fig. 9 Decomposition of a Multicast Tree into a set of unicasts and fanouts

4.4 The MCT Development Scenario

Shortly, the process could be seen as follows: the MultiCast Agent (MCA) clones itself and places it’s embryonic object in every node of the precalculated MCT in a fanout by fanout mode. Each embryonic multicast agent initially has only one (unicast) interface to communicate with the parent MCA (i.e with that MCA instance which had created it). During the MCT development each copy agent activates needed number of copies of needed interfaces. From the receiver viewpoint there is no difference to communicate with the remote MCA (placed at the multicast flow source) or with the MCA at the nearest splitting point. We assume that from the functional viewpoint each copy of the MCA provides the same service for the receiver[s].

If the MCA would be provided at all MCT nodes, including those which had no initially attached dedicated receivers, then when obtaining the *join* request they can simply invoke needed interface.

What we have as the result of the MCT development process is a tree spanning all confirmed participants of the group; each being provided with the needed multicast functionality. In terms of a multicast flow distribution this tree could be seen as a hierarchy; however from the multicast call establishment viewpoint this tree is a distributed association of peer agents. These agents share the multicast flow being sent to a group by the source, however their relation to receiver hosts are the same as described for Fig. 9.

The embryonic MCA is shown in Fig.10. Within the object-oriented programming termi-

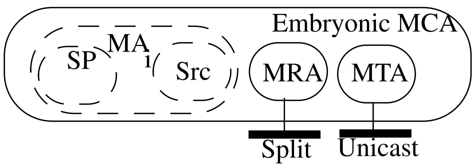


Fig. 10 Object presentation of the Multicast Agent

nology [37] the Embryonic MCA could be seen as a complex object with SP, MTA. MRA, Src being it's methods, however we'll call them also objects with their own methods (only *split* and *unicast* are shown here). The methods are invoked by messages according to the multicast call establishment scenario.

When the multicast call traverses several domains (where domain could be represented by a single service provider and/or network management) then at the border node of a new domain the MA has to invoke it's SP method to renegotiate all needed parameters and services (media, etc.). At the same time the whole new domain is to be visible to the source domain as a single receiver, that's why the border node within the sender domain has to invoke the MRA method and to communicate with the Src method of the MA copy inside a new domain (Fig. 11):

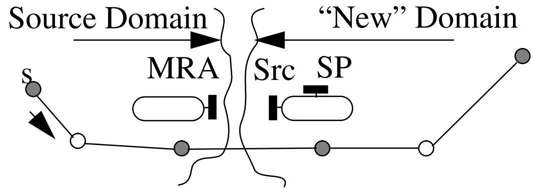


Fig. 11 Multidomain Operation

4.5 Object Framework Description

The complete set of objects introduced to support the object-oriented framework for a scalable multicast is presented in Figure 12. This set is called a framework because it is not elaborated here in the sense of object messages, also some methods are optional and others could arise during the multicast implementation. These building blocks can support the whole lifecycle of the multicast session, including call establishment and call modification.

The call establishment phase needs to invoke the Src and SP objects to establish primary contract with the service provider and MTA object to clone, transmit and self-install the MA complex object within the nodes of the MCT. Note that we suppose to clone not an abstract MCA complex object but it's instance customized according to a pre-negotiated set of session parameters (dedicated receivers list; media list; default QoS parameters; maximum number of active senders, etc.). The call establishment phase continues with invoking of MRA objects and their local negotiations with local receivers, session advertising, etc.

The call modification phase can start even during the Call Establishment. For example some of the dedicated receivers could become unreachable; crossing the domain boundary may cause changes in the default session parameters, one of possible active senders may ask for a floor even before the start of the initial flow, etc. We assume that this could not cause any problem because any MCA instance after the self-installation is ready to accept any of the messages which could take place within the entire multicast session. The shortcoming of this approach is that we have no opportunity to change all global parameters of the multicast session (DR list, Media List and default QoS for the reserved Media). The only possibility to change the DR list is to apply to the initial source.

Note, that if the number of dedicated receivers is too large for a service provider to permit them to send in parallel, then the number of active sources is to be introduced. However this number is to be somewhat less than the actual possible number of concurrent sources because we assume no information exchange between the instances of the MCA. Consequently the only way for them to know the number of actual active sources is to listen to the multicast. However, due to a delay introduced by the network several sources could start sending simultaneously based on their local permission. To prevent this situation we assume simply that the MA instance nearest to the last source has to stop sending when recognizes that actual number of active sources is greater than default maximum.

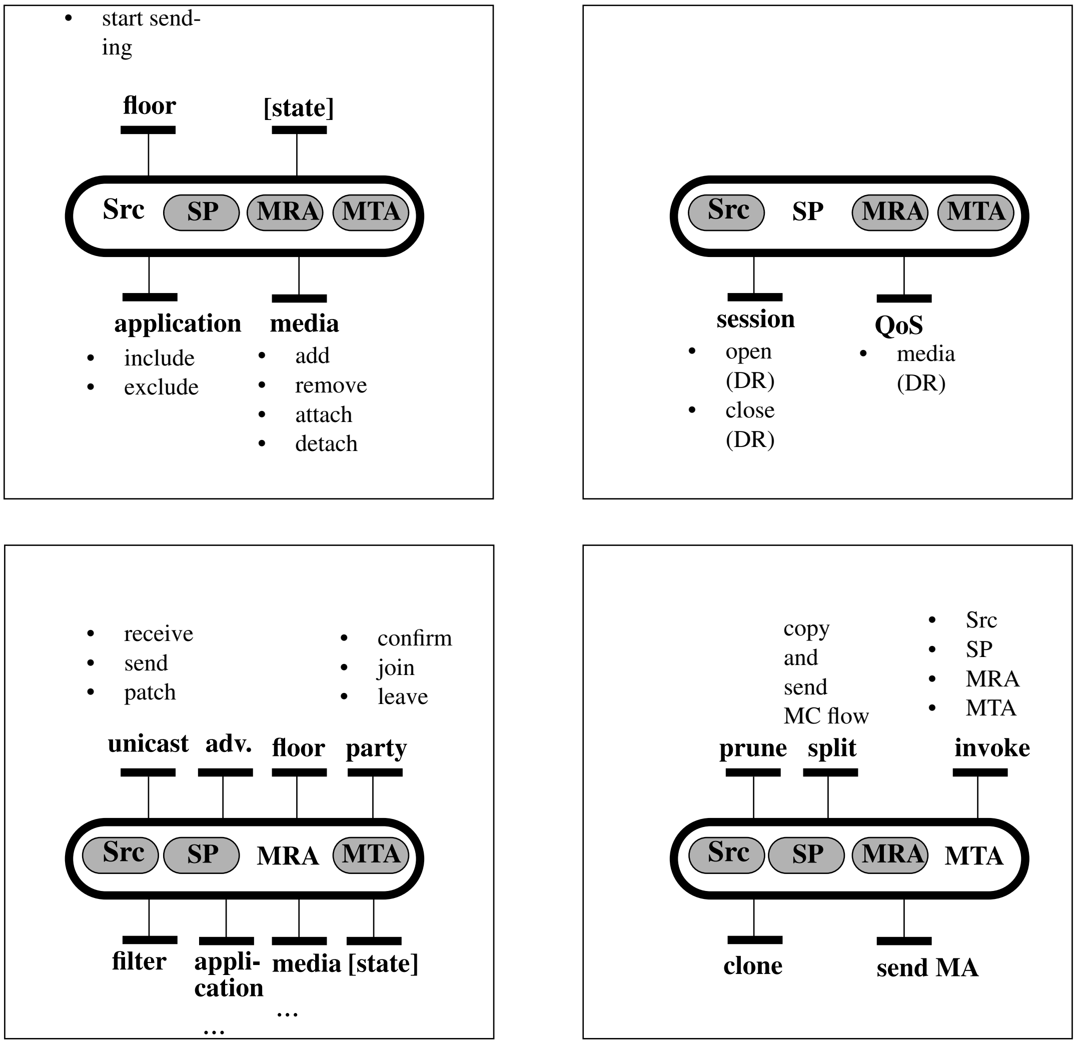


Fig. 12 The Object Framework

4.6 The MCTD Algorithm

The behaviour of the splitting node router processing the MCT development with the use of the multicast agent is presented in Figure 13 with the use of the timed coloured Petri Net.

The timed coloured Petri net is a place - transition net with coloured tokens representing some attribute[s] of the modelled information flow. The time delays and some data structures are associated with the positions; while transitions represent some rules for the tokens' attributes processing. TCPN are used here because of the two advantages: their expressive richness and ability for analytical study under some structural requirements [6, 7].

In Fig. 13 the following positions are used. Position `contr` represents the initial contract (transported within the *embryonic MCA* across the network) which initializes the current router with the following contract structural parameters regarding the current node (current fanout):

- the number of the end receivers R (each being the MCT leaf d-node) attached to the current node, placed in position `R`;
- the number of advertised routers A, placed in position `A`;
- the number of splitting nodes S, placed in position `S`;
- information about advertised routers which could be used as alternative splitting nodes, placed in a position `A/S`.

The latter relation between splitting nodes and their possible alternatives is given with the table T, the same symbol is used to mark corresponding arcs in the Petri net. Each splitting node of the fanout (if any) may have from zero to several alternatives in this table. The case of several overlapping alternatives is not considered in this paper, however all existing alternatives are assumed to be prioritized during the MCT construction, which function belongs to the service provider. Simultaneously with the placement of needed sets (`{r}`, `{a}`, and `{s}`) of adjacent node's identifiers in corresponding positions, the needed amount (`R+S+A`) of MCA copies is prepared and placed in the position `mca copy`.

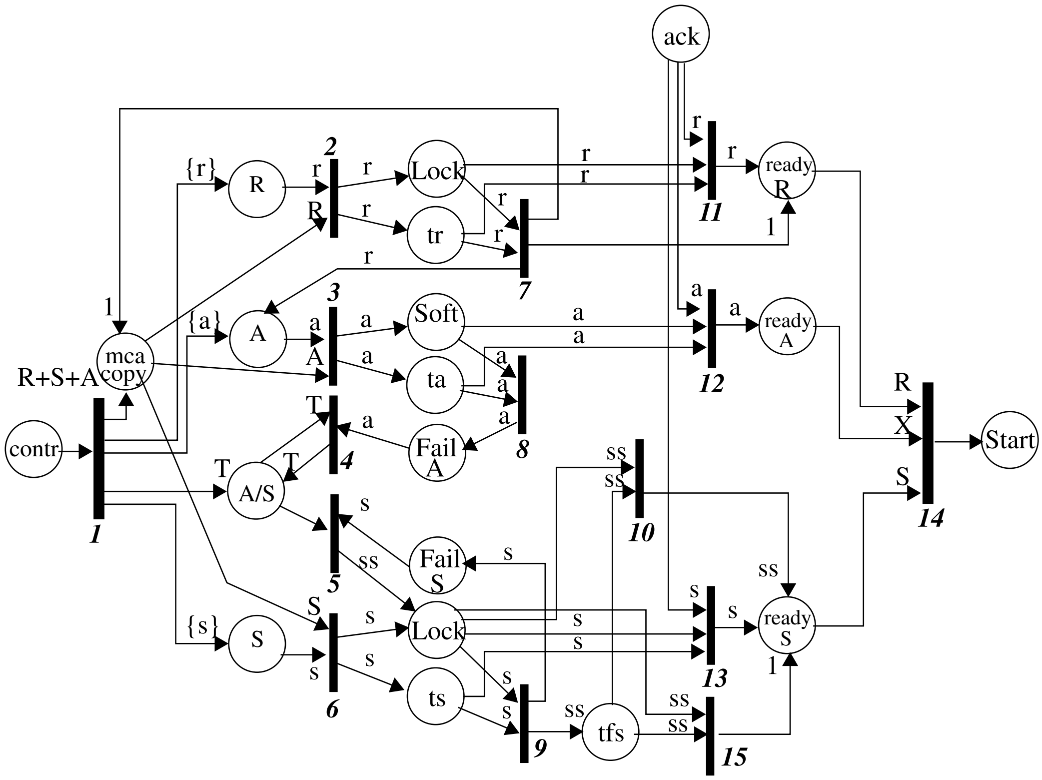


Fig. 13 The splitting point router behaviour model

The rest of the diagram presents the fanout development concerning one splitting point. The corresponding router behaviour has three distinguishable parts: sending MCA copies and receiving some negative or positive acknowledgements from

- advertised routers (sending MCA to these routers is an optional feature, however it is considered in this paper that the best alternatives to existing splitting points could be found within advertised routers[1]);
- end receivers; and
- splitting points.

Let us consider these 3 cases.Transition ***2*** models sending a MCA copy to every end receiver (`r`); after sending this receiver is considered locked upon receiving positive acknowledgement (positions `Lock` and `tr`). If a positive acknowledgement is received (transition ***11***) then receiver `r` is considered ready to accept the multicast flow (position `ready R`), corresponding tokens are removed from `Lock` and `tr`; if timeout `tr` expires before receiving the acknowledgement from `r` then (transition ***7***) the end receiver `r` is considered further as advertised - tokens `1` and `r` are moved respectively to `mca copy` and to `A`. That means the router `r` may join the multicast session later, however a simple token (`1`) is placed in `ready R` for proper counting of re-

1. This is not necessarily the case; for some network graphs the MCT construction algorithm may provide better alternatives within the set of end-receivers or within both sets. The splitting point router behaviour in any of these cases is easily derived from a generic model presented here.

ceivers.

Transitions (***3***, ***12***) and (***6***,***13***) as well as positions (`Soft`, `ta`, `ready A`) and (`Lock`, `ts`, `ready S`) represent the same successful scenario for devitalized routers and splitting points respectively. However if the advertised router is silent during the timeout period `ta` (transition ***8***) then the table `T` for possible alternatives for splitting points has to be updated (transition ***4***). Consequently, if splitting point `s` fails (position `Fail S`) then, probably updated, table `T` is used to find the best alternative to which the copy MCA is to be sent, however, now with the different modality (transition ***5***). Simultaneously with the locking of this substituted splitting point `ss` a new timer `tfs` is set up to wait for this additional MCT development. Note that the MCA copy could be installed already in `ss` (however as for advertised router) now this router has to elaborate the MCT further according to the MCT part description sent to it as a part of the contract. If `ss` fails to fulfil the job before the `tfs` expires (transition ***15***) then a simple token is sent to `ready S` position. The transition ***14*** represents a decision making on whether to start the multicast session or not. This decision is made on the base of any number (`X`) of ready to participate devitalized routers, of `R` ready end receivers and of `S` splitting points (where some of the receivers and splitting points could be marked by simple tokens as failed).

5. Implementation Remarks (Future Research)

Future research areas for the MCTC 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. 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.

The following implementation approach seems to be possible for the MCTD.The technical availability of the migrating multicast agent could be explained within the CORBA specification as shown in Figure 14.

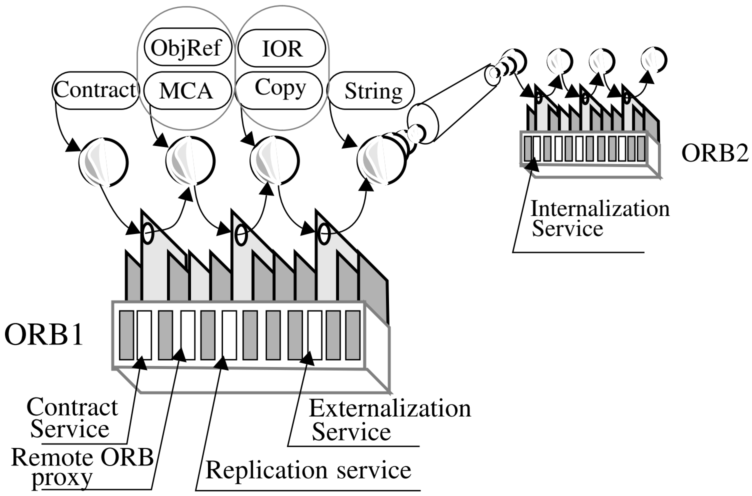


Fig. 14 Migrating MCA in CORBA

To solve the problem of unique object references the CORBA suggests [2] the mechanism of the interoperable object reference (IOR) externalization together with the externalization of the copy object; consequently two proxies are to be installed within the naming services of two collaborating ORBs. This paper suggests additionally that no reference is to be created for a copy MCA except the IOR. Moreover suggested locking of the destination actually locks the copy MCA as well, preventing from confusion with multiple objects within one ORB.

The CORBA specification was chosen among others for the following reasons:

- CORBA implementations provide for efficient interoperability across heterogeneous software platforms;
- being a software development platform it has an application viewpoint;
- CORBA deployment is rather fast in the direction for a cross-compatibility with ODP (TINA) [12]; OpenDoc, OLE [13] and CAD's Portable Common Tool Environment [14];
- adopted UNO approach [2] enables flexible distributed systems architecture.

The current approach was elaborated within the STEP group of GMD FOKUS [15] as a pre-project proposal aiming at highly scalable and reliable mechanism for application integration via multicast.

Future research in this area will consider quantitative scalability evaluation via different modelling approaches, as well as functional extensions of the presented framework. Clearly the proposed algorithms could be beautified in many directions.

References

1. Start K, Patel A. *The Distribution Management of Service Software*. - Computer Standards and Interfaces, 17(1995):291-301

2. Winter P.: Steiner Problem in Networks: A Survey. *Networks*, vol.17(1987), pp.129-167

3. Bhagwat P., Mishra P.P., Tripathi S.K.: *Effect of Topology on Performance of Reliable Multicast Communication*. In: *IEEE INFOCOM'94,* pp. 602 - 610

4. 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

5. 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

6. 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

7. Kou L., Markowsky G., Berman L., *A Fast Algorithm for Steiner Trees*. Acta Informatics, Vol. 15: 141 - 145, 1981

8. Plesnik J. *Heuristics for the Steiner problem in graphs*. Discrete Applied Mathematics, 37/38 (1992): 451 - 461, North-Holland.

9. Deering S., Estrin D., Farinacci D., Jacobson V., Liu Ch.-G., Wei L. *An Architecture for Wide-Area Multicast Routing,* In: Proc. SIGCOMM 94 -8/94, pp.126 - 135

10. Mitzel D.J., Estrin D., Shenker S., Zhang L. : *An architectural comparison of ST-II and RSVP*, In: *IEEE INFOCOM'94*, pp. 716 - 725

11. 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

12. Deering S. *Host Extensions for IP Multicast*, RFC 1112, August,1989.

13. Deering S. (Ed.) *IGMP Router Discovery Messages*, RFC 1256, September, 1991

14. Deering S., Estrin D., Farinacci D., Jakobsen V. *IGMP Router Extensions 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

15. Moy J. *Multicast extensions to OSPF*, RFC 1584

16. Marlow D. *Host Group Extensions for CLNP Multicasting*, RFC 1768, March,1995.

17. *Support for Multicast over UNI 3.1 based ATM Networks.* Internet draft: draft-ietf-ipatm-ipmc-04.txt.

18. Cheung S. Y., Kumar A.: *Efficient Quorumcast Routing Algorithms*, In *IEEE INFOCOM'94*, pp. 840 - 842

19. Schacham N., Meditch J.S. *An Algorithm for Optimal Multicast of Multimedia Streams*,

In IEEE INFOCOM'94, pp. 856 - 864

20. Ammar M.H. *Probabilistic Multicast: Generalizing the Multicast Paradigm to Improve Scalability.* In: *IEEE INFOCOM'94*, pp. 848 - 855.

21. 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

22. R. Perlman: *Interconnections. Bridges and Routers*. Addison-Wesley; 1992, 389 p.

23. 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

24. Heijenk X.H., Niemegeers I.G. *Communication Systems Supporting Multimedia Multi-user Applications.* - IEEE Network, January/February 1994, pp.34 - 44

25. Minzer S., Bussey H., Porter R., Ratta G. *Evolutionary Trends in Call Control.* - Proc. XV International Switching Symposium, ISS'95. Vol.2, pp. 300-304

26. Maastricht van C., Schalk E. *Call modelling in a Broadband IN architecture*. - In. Proc. XV International Switching Symposium, ISS'95. Vol.2, pp. 340 - 344

27. Grebenö J., Hanberger N., Ohlman B. *Experiences from implementing an Object Oriented Call Model.* - Proc. XV International Switching Symposium, ISS'95. Vol.2, pp. 77 - 81

29. Murata S., Shionozaki A., Tokoro M. *A Network Architecture for Reliable Process Group Communication.* - In Proc. 14th Int. Conf. on Distributed Computing Systems, June, 1994, Poznan, PL, ICDCS'94 (Piscataway, N.J.: IEEE), pp. 66 - 73

30. Covaci S. *Towards the Global Provision and Management of a New Open Bearer Service*, In: Proceedings of the First International Workshop on High Speed networks and Open Distributed Platforms (HSN&ODPl), June, 1995, St.Petersburg, Russia, 16 p.

32. Doar M., Leslie I. *How Bad is Naïve Multicast Routing ?* In Proc. INFOCOM'93, pp. 82-89.

33. 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.

34. Smirnov M.I. *Efficient Multicast Routing in High Speed Networks.* - In: Proceedings of the First International Workshop on High Speed networks and Open Distributed Platforms (HSN&ODPl), June, 1995, St.Petersburg, Russia, 17 p.

35. Casner S. *The MBONE Frequently Asked Questions,* at: http://www.research.att.com/mbone_faq.html

37. Prabhat A. K., Gretzinger M.R. *Distributed Object-Oriented Data-Systems Design*, Prentice Hall.

38. Casti J. *Connectivity, Complexity and Catastrophe in Large-Scale Systems*, Wiley, 1979.

39. Magedanz T., *Intelligent Agents - State of the Art and Potential Application Areas in Future Telecommunications*. - In: Proceedings of the First International Workshop on High Speed networks and Open Distributed Platforms (HSN&ODPl)/Ed. V. Tschammer, M. Smirnov, June, 1995, St.Petersburg, Russia, 17 p.

40. Luotonen A., Altis K. , "*World-Wide Web Proxies*," in First International WWW Conference, (Geneva, Switzerland), May 1994, 8 pp.

41. Gajewska H., Kistler J., Manasse M.S., Redell D.D., "*Argo: A system for distributed collaboration*". - In ACM Multimedia, ACM, 1994.

42. ORB 2.0 RFP Submission "*Universal Networked Objects*", Sept.,1994, OMG TC Document 94.9.32

43. "*Comparison of the OMG and ISO/CCITT Object Models. The Report of the Joint NM Forum/OMG Taskforce on Object Modelling*", OMG Document 94-12-30, 27 p.

44. Adler R.M., *Emerging Standards for Component Software*. - In Computer, March, 1995, vol.28, No. 3, p. 68-77

45. *Relationship of PCTE to OMA*, Revision 1.0, OMG TC Document 93.4.7, 23.04.93
46. *Home page of STERP group in FOKUS*, http://www.fokus.gmd.de/step/entry.html

Appendix (Numerical example)

To give an impression of how the MCTC algorithm works let us apply it to the graph depicted in Figure A1.

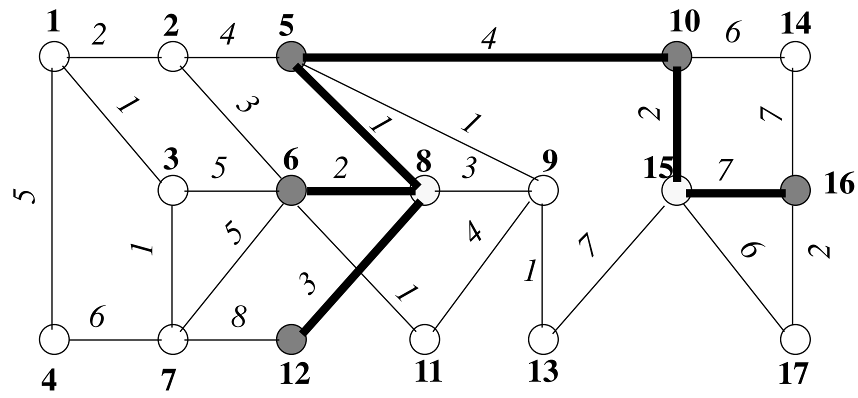


Fig. A1 Sample network graph

The resulting multicast connectivity is shown in bold edges of the graph. Initial graph presentation (D1) is given in Table 4 and resulting data structure (D2) depicted in Table 5

Table 4:

5	8	9				2	10							
6		11	8		2			3	7					
10			15				5			14				
12					8									7
16			17									14	15	
1		3	2						4					
2			1		6		5							
3	1	7							6					
4									1	7				
7		3							6	4				12
8		5	6	9	12									
9	5	13		8			11							
11		6					9							
13		9											15	
14										10			16	
15			10								13	16	17	
17			16										15	
Cost	1	1	2	3	3	4	4	5	5	6	7	7	7	8

Table 5:

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

MCTC and MCTD operation over the sample networ are given in steps in Figures A2 and

MCTC Algorithm:

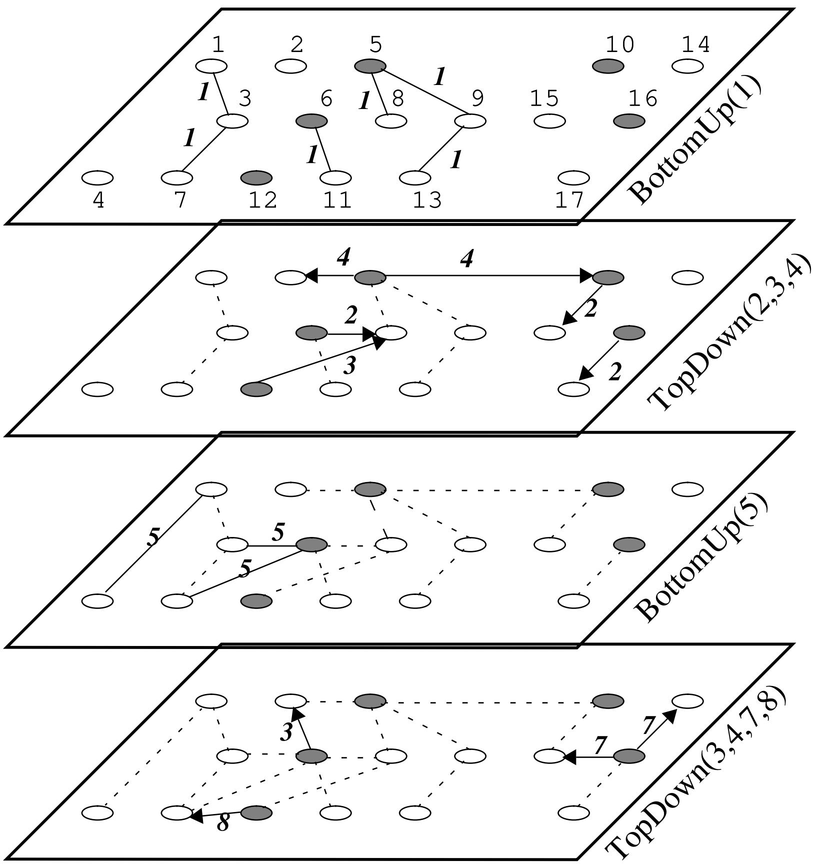


Fig. A2 The MCT Construction

A3 respectively.

MCTD Algorithm:

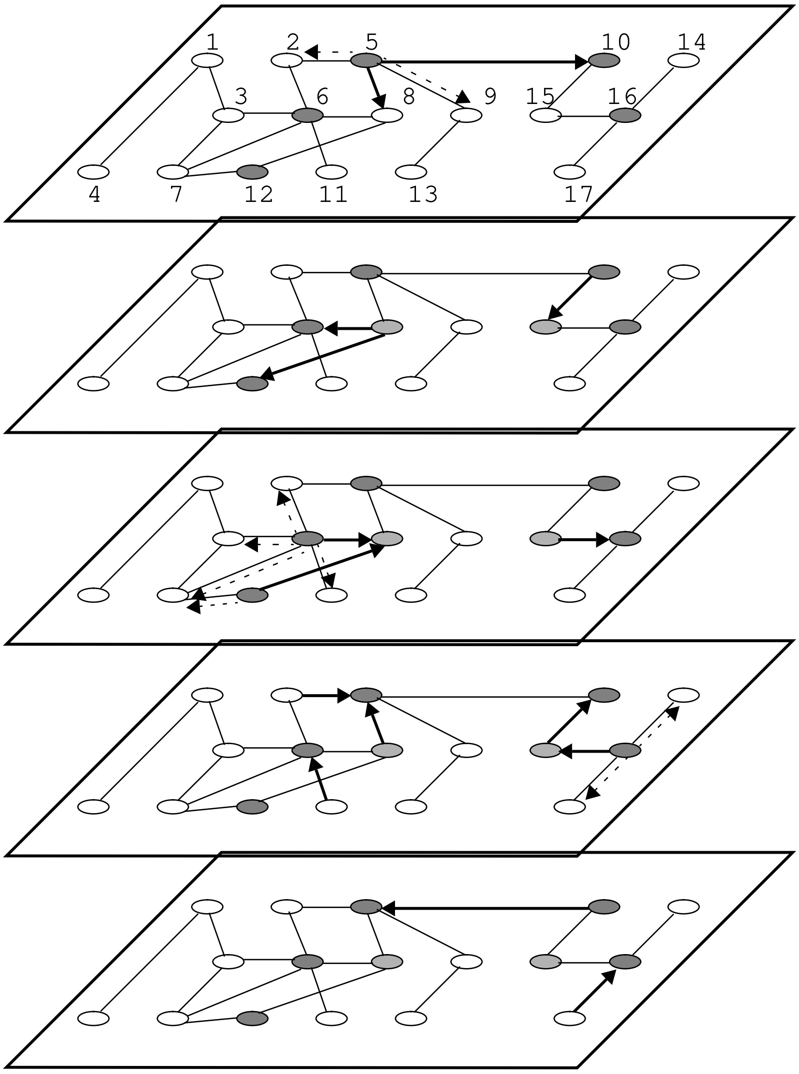


Fig. A3 The MCT Development