A NEW ALGORITHM FOR CONSTRUCTION OF A P2P MULTICAST HYBRID OVERLAY TREE BASED ON TOPOLOGICAL DISTANCES

Sergej Alekseev[1] and Jörg Schäfer[2]

[1]Department of Computer Science and Engineering, Computer Networks and OS,
Frankfurt University of Applied Sciences, Germany
alekseevf@fb2.fra-uas.de
[2]Department of Computer Science and Engineering, Distributed Systems,
Frankfurt University of Applied Sciences, Germany
jschaefer@fb2.fra-uas.de

ABSTRACT

In the last decade Peer to Peer technology has been thoroughly explored, because it overcomes many limitations compared to the traditional client server paradigm. Despite its advantages over a traditional approach, the ubiquitous availability of high speed, high bandwidth and low latency networks has supported the traditional client-server paradigm. Recently, however, the surge of streaming services has spawned renewed interest in Peer to Peer technologies. In addition, services like geolocation databases and browser technologies like Web-RTC make a hybrid approach attractive.

In this paper we present algorithms for the construction and the maintenance of a hybrid P2P overlay multicast tree based on topological distances. The essential idea of these algorithms is to build a multicast tree by choosing neighbours close to each other. The topological distances can be easily obtained by the browser using the geolocation API. Thus the implementation of algorithms can be done web-based in a distributed manner.

We present proofs of our algorithms as well as practical results and evaluations.

KEYWORDS

Distributed algorithms, peer-to-peer (P2P), hybrid, overlay multicast tree, live streaming

1. INTRODUCTION

Peer-to-Peer (P2P) streaming has become more and more popular nowadays again after interest in general P2P has generally decreased after the initial enthusiasm in the late 90 – partially due to the ubiquitous and quick availability of high speed, high bandwidth and low latency networks which has supported the traditional client-server paradigm in the last decade. The central strength of P2P streaming systems is the capability of sharing resources so that larger (and more costly) servers can be replaced by smaller (and cheaper) computers. The P2P networks are build usually as a logical overlay network. The contribution of this paper is the construction and management of a P2P multicast tree streaming overlay where the nodes are physically close to each other in

Jan Zizka et al. (Eds) : CCSIT, SIPP, AISC, CMCA, SEAS, CSITEC, DaKM, PDCTA, NeCoM - 2016
pp. 307–321, 2016. © CS & IT-CSCP 2016 DOI : 10.5121/csit.2016.60126

the underlying network. In this paper we present two algorithms. The first is the joining algorithm that each node runs when it enters the system. The essential idea of the algorithm is to construct a multicast tree structure by finding a suitable neighbour in the overlay multicast tree and considering resources of peers. The second algorithm handles a host leaving that occurs gracefully or accidentally. For both algorithms we provide full mathematical proofs of minimality features. In addition, we present some experimental results and evaluations. And finally we conclude our paper with remarks on possible future work.

2. RELATED WORKS

In recent years a number of P2P-based applications for stream delivery have been developed – e.g. Zattoo (http://zattoo.com), PPTP (http://www.pptv.com) and Octoshape (https: // octoshape. com).

To improve the scalability and to optimise the usage of resources in the P2P network, several approaches have been proposed. In [1] various problems that arise due to the fact of P2P systems being highly dynamic and heterogenous are examined. It focuses especially on resilience mechanisms. In [2] and [6] an overview of application and network layer mechanisms are presented and the Mesh and Multiple-Tree P2P overlays are compared.

Several applications have been developed for various categories of mesh based P2P streaming. The authors of [8] and [7] present a hybrid approach for overlay construction and data delivery in an application-layer multicast. The HyPO approach in [7] optimizes the overlay by organizing peers with similar bandwidth ranges in a geographical area into a mesh overlay. The ToMo approach in [7] combines the strong points of a tree-based structure and a mesh-based data delivery to a two-layer hybrid overlay. The mTreebone of [9] is a collaborative tree-mesh design that leverages both mesh and tree structures. The key idea is to identify a set of stable nodes to construct a tree-based backbone with most of the data being pushed over this backbone. AnySee [5] is a mesh based P2P system in which resources are assigned based on their locality and delay.

In the present work we propose algorithms to construct a tree based multicast overlay based on topological distances. Similar approaches are described in [12], [3] and [14]. Already in [20] an architecture has been proposed for designing a global internet host distance estimation service. However, only relatively recently geographical information has become practically available from freely available geolocation databases [16], and therefore ideas which have been of theoretical value only have now become practical, see also [19]. The approach used in [12] and [3] organizes the peers into a hierarchy of clusters such that the neighboring peers are grouped into the same cluster. The overlay network is build from the cluster leaders to the other members recursively. In [14] a locality-aware P2P overlay construction method, called Nearcast, is proposed which builds an efficient overlay multicast tree by letting each peer node choose physically closer nodes as its logical children. Whereas there is rather comprehensive coverage of theoretical P2P algorithms and mathematical theorems on some of them like e.g. the T-Man protocol, see [4], up to our best knowledge, no minimality results have been proven for the overlay networks like the one described above but rather simulation results have been computed. In our work we propose algorithms which minimise the routing costs, usage of peer resources and end-to-end delay based on the topological location of peers. We provide a proof for the minimality of routing costs and provide evidence for keeping end-to-end delay low.

3. PROPOSED APPROACH

The concept of P2P multicasting [11], [12] is often applied to reduce the costs needed to deploy and to maintain services related to streaming of various content to many users, e.g. VoD, IPTV, radio, news channels, etc. In this paper, we propose an approach to the construction of a P2P overlay multicast tree with the goal to solve the following important problems:

– **Optimal routing between peers:** Transmission at an overlay P2P-network might be inefficient, especially when the P2P-network is randomly constructed. This stems from the fact that the distance between peers physically or topologically is not considered by constructing the P2P-network.

– **Optimal usage of peer resources:** Peer resources include available bandwidth, processing power and storage space.

– **End-to-End delay:** The end-to-end delay is the latency, accumulated peer by peer, for the delivery of a data packet along the overlay path from the source host to an end host. To reduce this delay the height of the multicast tree should be kept small.

– **Handling of peer connections:** In practice the P2P-network need to deal with peers joining the network and peers that leave voluntarily or due to failure.

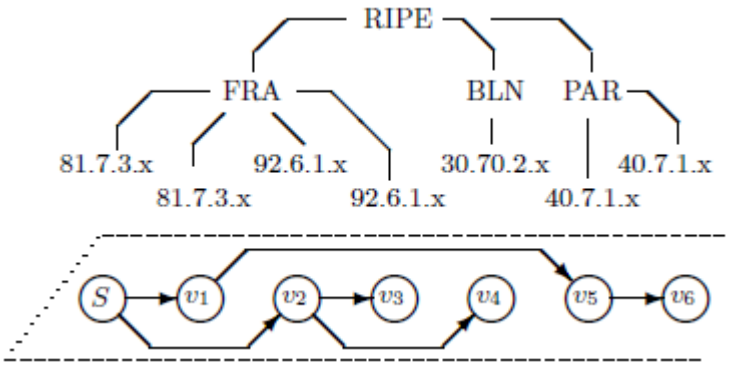

Figure 1. Topological search tree and p2p multicast tree structure.

To overcome these problems, we propose algorithms for the construction and the management of an overlay P2P-network. Our algorithms use the topological distances between peers to guarantee the optimal routing costs. We define two data structures, a topological search tree and a P2P multicast tree (fig. 1). The search tree is used to find the nearest peer to be attached to the multicast overlay. This a special case of the Nearest Neighbour Search (NNS) or closest point search problem. Donald Knuth named this problem the post office problem [10]. The problem relates to an application of the assignment to the next post office. In our case the problem is reduced to the search in the tree and adapted for the search of an optimal usage of peer resources. The P2P multicast tree is used for the actual data transfer.

4. P2P OVERLAY MULTICAST TREE CONSTRUCTION AND MAINTENANCE

4.1. Definitions and Preliminaries

To identify the topological position of hosts in a network, a unique H-dimensional coordinate C is assigned to each host. The idea to use the network coordinates is based on considerations from [13], [14] and [15]. In contrast to the algorithms presented therein, we use in our approach two data structures: the search tree T_s for searching the nearest neighbour according the topological position in the network and the multicast tree T to connect hosts to a P2P overlay multicast network.

The multicast overlay tree is defined as T = (V, E), where V is a set of vertices, which represent the end hosts, and E is a set of directed edges, which represent data delivery streams between the end hosts.

The search tree T_s is considered as an H-layered topological tree. According to the topology of the search tree T_s for each vertex $v \in V$ the *network coordinate* $C(v)$ is defined as follows:

$$C(v) = \{C_{H-1}(v), ..., C_0(v)\} \tag{1}$$

Similarly to the Nearcast method proposed in [14] we use static *network coordinates* and assign to the vertices (end hosts in the physical network) special geographical meanings. The coordinates $C_{H-1}(v), ..., C_0(v)$ represent *Regional Internet Registry*, *Country*, *City* and *n-th* bits of an *IP*-address respectively e.g:

$$\{RIPE, DE, HH, 80.x.x.x, 80.6.x.x., 80.6.60.x\}$$
$$\{ARIN, US, NIC, 10.x.x.x, 10.7.x.x., 10.7.50.x\}$$

The geographical information can be easily obtained from the freely available geolocation databases [16] by using the programming interfaces described in [17] and [18].
Formally the search tree T_s can be defined using tuple notation as $T_s = (V_s, E_s)$, where

$$V_s = \bigcup C(v) \tag{2}$$

and

$$E_s = \bigcup_{0<i<(H-1)} \{(C_i(v), C_{i+1}(v))\}. \tag{3}$$

Finally we introduce a hierarchical common network distance D and last common coordinate LCC, used by our algorithms. The hierarchical common network distance D between two vertices v_x and v_y with the static network *coordinates*:

$$C(v_x) = \{C_0(v_x), ...C_i(v_x)...C_{H-1}(v_x)\}$$
$$C(v_y) = \{C_0(v_y), ...C_i(v_y)...C_{H-1}(v_y)\}$$

is the number of coordinates with different values and is denoted as $D(v_x, v_y)$. Formally the hierarchical common network distance is defined as:

$$D(v_x, v_y) = H - 1 - m$$
$$m = \max_{0 \le i \le H-1}\{i \mid C_k(v_x) = C_k(v_y)\ \forall k \le i\} \quad (4)$$

e.g. for the following vertices

$$v_x = \{RIPE, DE, FRA, 80.x.x.x, 80.70.x.x\}$$
$$v_y = \{RIPE, DE, BLN, 90.x.x.x, 90.80.x.x\}$$

the hierarchical common network distance is $D = 3$.

The last common coordinate LLC of two vertices is the last identical coordinate in the order of $C_0, C_1, \ldots C_i$, formally:

$$LCC(v_x, v_y) = C_i \iff C_k(v_x) = C_k(v_y) : 0 < k < i \quad (5)$$

In the example above the $LCC(v_x, v_y) = DE$.

4.2. Joining algorithm

To construct a multicast overlay tree the joining algorithm connects the hosts to an overlay network by analysing the geolocation information provided by the end hosts. The algorithm can be implemented in a centralized or a distributed manner. The pseudocode of the joining algorithm is shown in fig. 2.

Algorithm *JOIN* (new, T_s)

1 $T_s = T_s \cup C(new)$
2 find the nearest neighbour n of *new* in T_s;
3 attach *new* to n;

Figure 2. The pseudocode of the JOIN algorithm

Figure 3 illustrates an example of the joining nodes to an existing overlay network. Initially the overlay multicast tree contains only a source host S and the search tree T_s includes the coordinates C(S) (fig. 3a). To attach the new node v_1 the joining algorithm extends the search tree T_s by the adding the coordinates $C(v_1)$ and determines the nearest host by traversing the search tree Ts. The nearest neighbour can be easily found by a simple tree traversing in O(*log n*) time. The new host v_1 is attached to the host S (fig. 3b). The fig. 3c illustrates the attaching of the host v_2 to the multicast overlay tree.

To show that the routing in the constructed multicast tree T is optimally organised, we assign to each edge $e = \{v_0, v_1\}$ a topological distance value $D(e) := D(v_0, v_1)$, that represents the routing costs between the vertices v_0 and v_1. It is easy to check that D satisfies all axioms of a metric which is important for the minimality results presented in the sequel (only the triangle inequality is non-trivial). The sum of all distances $S(T) = \sum_{e \in E} D(e)$ is the total routing costs in the tree. The lower the value $S(T)$ is, the less

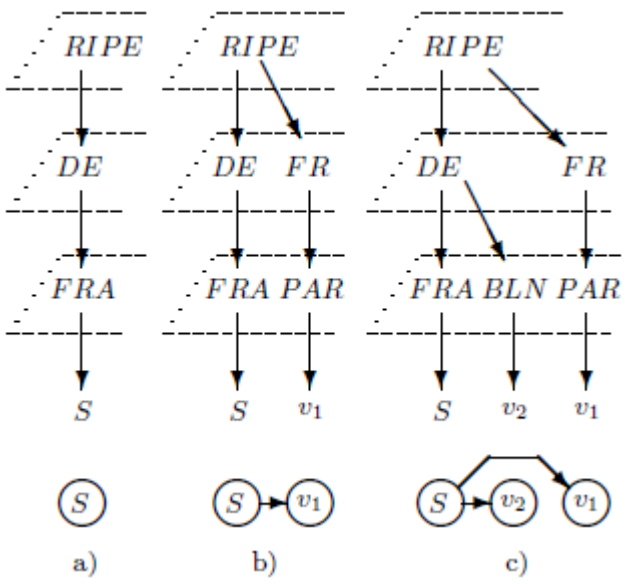

Figure 3. An example of the algorithm *JOIN* execution.

routing overhead is necessary to deliver the content to each vertex in the multicast tree. The topological distance can be thought of a proxy for the "real" distance measured as End-to-End-Delay or other QoS parameters. In the literature (see e.g. [21]) it has been argued, that the topological distance is a reasonable proxy in practice. As next we show that our algorithm constructs a multicast tree T with minimal routing costs. In other words it is not possible to construct another multicast tree T_1 with $S(T_1) < S(T)$.

Theorem 1. *The algorithm JOIN (fig. 2) constructs a tree T with the minimal $S(T)$ value.*

Proof. The correctness of the algorithm is proved by induction on the number of vertices in T.
Base case: $T = \emptyset$ or $|T| = 1$ are trivially minimal.
Induction step: Assume that $S(T)$ is minimal for n connected vertices. Let v_{n+1} be the next vertex added to the multicast tree T and $v \in T$ is the nearest neighbour of v_{n+1} (fig. 4a). Let us show that $S(T) + D(v, v_{n+1})$ is minimal.
Consider any multicast trees T', where v_{n+1} not connected to v. We will see that there is no tree with $S(T') < S(T)$.

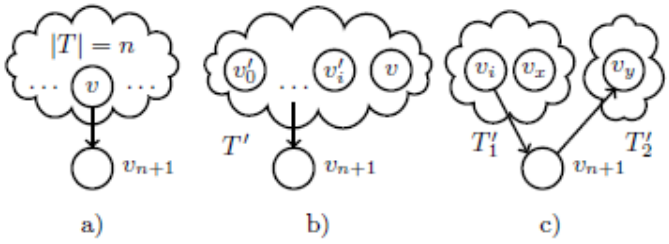

Figure 4. Proof by induction

First assume the vertex v_{n+1} is connected to any vertices $v' \in T \setminus v$ as a leaf (fig. 4b). It is easy to see that $S(T') \geq S(T) + D(v, v_{n+1})$, because $D(v', v_{n+1}) \geq D(v, v_{n+1})\ \forall v'$ as v is a nearest neighbour.

Thus, the $S(T')$ value could only be possibly reduced by connecting the vertex v_{n+1} such that an edge $e = \{v_x, v_y\}$ with $D(v_x, v_y) > D(v, v_{n+1})$ is removed from T (fig. 4c). Removing an edge from the tree breaks it into two separate subtrees T'_1 and T'_2. The vertex v_y is the root of the subtree T'_2, because it is the successor of the vertex v_x. In order to connect two subtrees the vertex v_y must be connected to the vertex v_{n+1} as a successor and the vertex v_{n+1} is connected to a vertex v_i in T'_1. It is possible that $v_i = v_x$ or $v_i = v_n$, if $v_n \in T'_1$.

With $T' = (T \setminus \{v_x, v_y\}) \cup \{v_{n+1}, v_y\} \cup \{v_i, v_{n+1}\}$, let us assume the following inequality being strict:

$$S(T') < S(T) + D(v, v_{n+1}) \tag{6}$$

The $S(T')$ value of T' can be calculated from the definition as:

$$S(T') = S(T) - D(v_x, v_y) + D(v_{n+1}, v_y) + D(v_i, v_{n+1})$$

As v has minimal distance, by replacing the $S(T')$ in the inequality (6) we get:

$$-D(v_x, v_y) + D(v_{n+1}, v_y) + D(v_i, v_{n+1}) < D(v, v_{n+1})$$

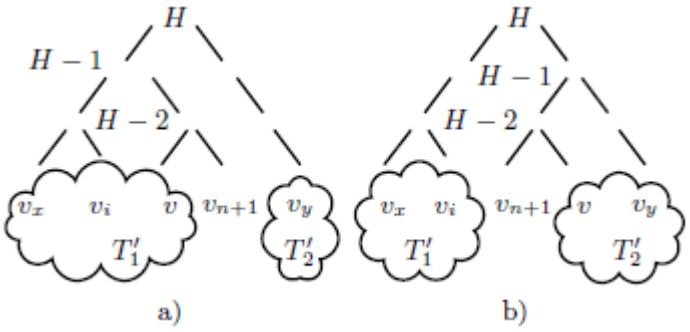

Figure 5. Topological distances

Thus, $D(v_x, v_y) > D(v_{n+1}, v_y)$ and $D(v_x, v_y) > D(v_i, v_{n+1})$.

However, from the definition of topological distances in the search tree T_s (fig. 5) the following must be true:

$$\begin{aligned}(D(v_x, v_y) = D(v_{n+1}, v_y)) \wedge (D(v_x, v_y) > D(v_i, v_{n+1}))\\ (D(v_x, v_y) > D(v_{n+1}, v_y)) \wedge (D(v_x, v_y) = D(v_i, v_{n+1}))\end{aligned} \tag{7}$$

But (7) contradicts inequality (6). Thus $S(T) + D(v, v_{n+1})$ is minimal. □

The algorithm *JOIN* (fig. 2) may construct different trees depending on selection of the nearest neighbour and the order the nodes joining, however the next theorem shows that $S(T)$ is not affected.

Theorem 2. *All trees constructed by the algorithm JOIN (fig. 2) have the same $S(T)$ value.*

Proof. Assume that algorithm $JOIN$ (fig. 2) constructs two different trees T_0 and T_1 for the same set of vertices (end hosts) with $S(T_0) \neq S(T_1)$.
According the theorem 1 the value of $S(T_0)$ and the value of $S(T_1)$ are minimal. Since $S(T_0) \neq S(T_1)$, follows that either $S(T_0)$ or $S(T_1)$ is not minimal. So the assumption must be incorrect. □

The algorithm $JOIN$ (fig. 2) solves the routing costs problem, mentioned in the section 3. However the algorithm does not consider the usage of peer resources. As next we present an extension of the algorithm $JOIN$ to solve the peer capacity problem.

4.3 Management of peer-resources

To manage the usage of peer resources the attribute *resource capacity* is assigned to each host in the network model. The *resource capacity* of an end host, denoted by $R(v)$, is a maximum number of outgoing links $e \in E$, which can be served by the vertex v. The value $R(v)$ is calculated based on available bandwidth and other resources of the peer.
The pseudocode of the joining algorithm with the peer-resource management $JOIN_R$ is shown in fig. 6 (we call v *in* $LCC(new, n)$ iff $LCC(new, v) = LCC(new, n)$). To join a new node the algorithm $JOIN_R$ finds the nearest neighbour n, similar to the algorithm $JOIN$ (fig. 2). Instead to attaching the node directly to the nearest neighbour n found, the algorithms checks all existing hosts with the same topological distance as the vertex n, whether one of the vertices has enough resources to forward the data link to the new

Algorithm $JOIN_R$ (new, T_s)

```
1  T_s = T_s ∪ C(new)
2  /* n is a potential neighbour of new */
3  for (all reachable hosts v in LCC(new, n)) {
4       if(R(v) != 0) {
5          attach new to v;
6          R(v) = R(v) − 1;
7          return;
8       }
9  }
10 for ( all reachable hosts v in LCC(new, n) ) {
11      for (all hosts v_x connected to v) {
12         if(D(v, new) ≤ D(v, v_x)) {
13            insert new between v and v_x;
14            R(new) = R(new) − 1;
15            return;
16         }
17      }
18 }
```

Figure 6. The pseudocode of the $JOIN_R$ algorithm

node. For that the last common coordinate LCC according the definition 5 (section 4.1) is calculated. If one appropriate host v is found the new node is attached and the resource capacity attribute of the host v is updated. Otherwise the algorithms checks again all reachable hosts and verifies if the new node can be inserted between a host v and any hosts connected to v with $D(new, v) \leq D(v_x, v)$.

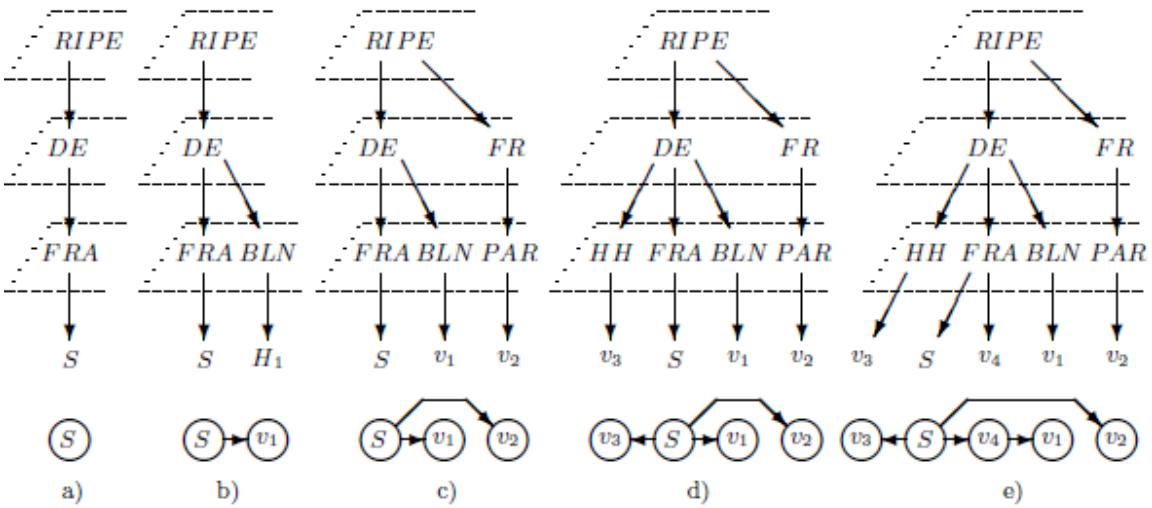

Figure 7. An example of the algorithm $JOIN_R$ execution

Figure 7 illustrates an example of the algorithm $JOIN_R$ execution. We define for this example the following condition $\forall v \in V : R(v) = 3$. In other words each host is able to maintain three outgoing links.

Initially the overlay multicast tree contains only a source host S and the search tree T_s includes the coordinates $C(S)$ (fig. 7a).
To attach the new node v_1 the algorithm $JOIN_R$ adds the coordinates $C(v_1)$ to the search tree T_s and determines the nearest host of v_1 (fig. 7b). The host v_1 is attached to the host S and the $R(S)$ value is updated accordingly $R(S) = 3 - 1 = 2$.
Figure 7c illustrates the attaching of the host v_2. After updating the search tree T_s the algorithm $JOIN_R$ checks all potential nearest neighbours, reachable from the $LLC = RIPE$. In order to find the LLC-value, it is enough to traverse backwards the search tree T_s from the vertex v_2 until the first branch. The potential nearest neighbours of v_2 are the hosts S and v_1, because $D(S, v_2) = D(v_1, v_2) = 2$. The host v_2 is attached to the host S and the $R(S)$ value is updated accordingly $R(S) = 2 - 1 = 1$.
The attaching of the host v_3 (fig. 7d) is similar to the previous step. The $R(S)$ value is updated to $R(S) = 1 - 1 = 0$. The host S can not maintain any further outgoing links.
The last figure 7e illustrates the attaching of the host v_4. The nearest neighbour of v_4 is S. But $R(S) = 0$ and there are no other free potential neighbours with the same topological distance. The algorithm $JOIN_R$ checks in this case all potential nearest neighbours v whether any hosts v_x with $D(v, v_4) \leq D(v, v_x)$ is connected to v. In our case:

$$D(S, v_4) = 0 \leq D(S, v_1) = 1$$
$$D(S, v_4) = 0 \leq D(S, v_2) = 2$$
$$D(S, v_4) = 0 \leq D(S, v_3) = 1$$

So the algorithm $JOIN_R$ inserts the host v_4 between S and v_1 and updates the $R(v_4)$ value accordingly $R(v_4) = 3 - 1 = 2$.
Similar to the algorithm $JOIN$ (fig. 6) we show that the routing in the constructed multicast tree T is optimally organised, i.e. that $S(T)$ is minimal *and* that the algorithm solves the resource capacity problem $R(T)$ as defined below. In order to do so we assign to each vertex $v \in V$ a resource value $R(v)$ that represents the maximum number of outgoing data links which can be served by the vertex. We call $R(T)$ *solved* iff $\forall v \in V : R(v) \leq R_{max}(v)$. Admittedly the algorithm constructs a multicast tree with minimal $S(T)$ value and solves the resource capacity problem $R(T)$ with respect to a following precondition:

$$\forall v \in V : R_{max}(v) > 0 \tag{8}$$

Theorem 3. *The algorithm $JOIN_R$ (fig. 6) constructs a tree T with the minimal $S(T)$ value and solves the resource capacity problem $R(T)$.*

Proof. The algorithm $JOIN_R$ consists of two parts, each one performing a loop on the potential nearest neighbours v of the host *new*.
The first part is reduced to the algorithm $JOIN$ (fig. 2) and proved by induction (theorem 1). If the first loop detects a nearest neighbour, then it must have at least one free outgoing link to attach a new vertex. So $S(T)$ is minimal and $R(T)$ is solved.
The second loop is only executed if all potential nearest neighbours have no capacity. According the precondition 8 each vertex must be able to serve at least one outgoing link. It follows that one of the potential nearest neighbours must be connected to a vertex x with the topological distance $D(v, x) \geq D(v, new)$ (fig. 8a). The vertex x is reconnected to the vertex *new* (fig. 8b). $D(v, x) = D(new, x)$, because v is one of the nearest neighbours of *new*. This step can be reduced to the algorithm $JOIN$ (fig. 2) and proved by induction (theorem 1). So $S(T)$ is minimal. According the precondition 8 the vertex *new* must be able to serve an outgoing link to x and R(T) is solved. □

In the next section we present an extension of the algorithm to reduce the end to end delay.

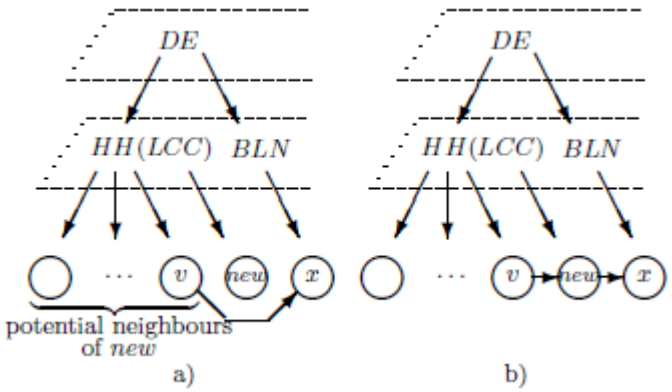

Figure 8. Proof of the second loop.

4.4 End-to-End Delay

An important performance metric that is of concern for live media streaming overlays is the End-to-End Delay.
The End-to-End Delay (EED) depends on the underlying network delay and the local delays at overlay peers due to queuing and processing. Formally we define the EED as a delay of the path $p = \{v_0, \dots v_n\}$ from the source host v_o to the end host v_n:

$$EED = \sum_{i=0}^{|p|} D(v_i, v_{i+1}) + |p| \tag{9}$$

We use the sum of the topological distances to represent the logical delay in the network and number of overlay hosts to represent the local delays at overlay peers. The algorithm $JOIN_R$ constructs an overlay multicast tree T by chosing topologically close peers as neighbours. The $S(T)$ value is minimal (theorem 3), so the delay in the network is minimal. To reduce total End-to-End Delay it is necessary to minimize the number of peers on the delivery path p. So the total End-to-End Delay depends on the multicast tree height.
In order to keep the End-to-End Delay small we adapted the algorithm $JOIN_R$ for construction a low stretched multicast tree as follows:

1. The loops on the potential nearest neighbours v of the host new are executed in the sorted order. The potential nearest neighbours are sorted by the End-to-End Delay to the source host according the equation (9).
2. The insert procedure (Fig. 9: lines 15-17) is modified. The host new is inserted between the host v and the host v_x with the lowest End-to-End Delay according to equation (9). And the outgoing links of v_x are reattached to new as long as $R(new) > 0$.

The pseudocode of the joining algorithm $JOIN_{RE}$ is shown in figure 9 (we call v in $LCC(new, n)$ iff $LCC(new, v) = LCC(new, n)$). Since the basic structure of the algorithm $JOIN_{RE}$ is equal to the structure of the algorithm $JOIN_R$, the algorithm satisfies the conditions of the theorem 1 and 3. For example in fig. 7c the potential nearest neighbours of the host v_2 are S and v_1. But the host S has the lower EED-value, so the host v_2 is attached to S. The host v_1 is attached to S (fig. 7d), because S has the lowest EED-value and has enough resources.
In fig. 7e the host v_4 can be inserted between three potential hosts v_3, v_1 and v_2. According to equation 9 the $EED(S, v_3) = 2$, $EED(S, v_1) = 2$ and $EED(S, v_2) = 3$. So the host v_4 is inserted between S and v_1.

The modified algorithm $JOIN_R$ keeps the End-to-End Delay small because the multicast tree height is logarithmic to the number of hosts. The total End-to-End Delay is $O(logN)$.

```
Algorithm JOIN_RE (new, T_s)

1  T_s = T_s ∪ C(new)
2  /* n is a potential neighbour of new */
3  for (all reachable hosts v in LCC(new, n)
                          in sorted order by EED ) {
4     if(R(v) != 0) {
5        attach new to v;
6        R(v) = R(v) − 1;
7        return;
8     }
9  }
10 for (all reachable hosts v in LCC(new, n)
                          in sorted order by EED ) {
11    for (all hosts v_x connected to v) {
12       if(D(v, new) ≤ D(v, v_x)) {
13          insert new between v and v_x;
14          R(new) = R(new) − 1;
15          while(R(new) > 0 or
                    v_x has outgoing links) {
16             reattach an outgoing link of v_x to new;
17          }
18          return;
19       }
20    }
21 }
```

Figure 9. The pseudocode of the JOINRE algorithm

4.5 Reconstruction algorithm

In order to support handling of peer connections we propose an algorithm to handle a host departure that may occur on purpose or by accident accidentally.
The pseudocode of the reconstruction algorithm is shown in figure 10. The algorithm deletes a host if it is a leaf. Otherwise it tries to reattach the outgoing links to the parent as long as it has enough resources. Finally the algorithm executes the algorithm $JOIN$ for remaining hosts.

5. EXPERIMENTAL EVALUATIONS

In this section, we present simulation results for evaluation of the proposed approach. We have implemented a simulation to create overlay multicast trees according to our approach and as a balanced binary tree. The simulation parameters are: number of hosts and the resource capacity of each host R. The simulation calculates the routing costs in the tree $S(T)$ according the definition in section 4.2, the height of the tree $H(T)$ and the End-to-End Delay (EED) according to the equation 9 (section 4.4).
The tables 1 and 2 present the results of the comparison a multicast tree constructed by the $JOIN$-Algorithm with a randomly constructed balanced binary tree with the $R(v) = 3$

```
Algorithm RECONSTRUCT_TREE (v_del)

1  if (v_del is a leaf) {
2        delete v_del;
3  } else {
4     v_p = parent of v_del;
5     for (each child v of v_del in sorted order) {
6        if(R(v_p) > 0) {
7           reattach v to v_p;
8           R(v_p) = R(v_p) − 1;
9        }
10       else {
11          JOIN(v, T_s );
12       }
13    }
14 }
```

Figure 10. The pseudocode of the reconstruction algorithm

Table 1. Comparison with $R(v) = 3$: JOIN - proposed algorithm, BBT - Balanced Binary Tree

# hosts	JOIN			BBT		
	$S(T)$	$H(T)$	EDD	$S(T)$	$H(T)$	EDD
10	13	3	4	17	4	10
100	103	5	7	180	10	20
500	503	6	8	850	14	28
1000	1003	7	9	1650	30	62

and $R(v) = 2$ accordingly. The results show the total routing costs in the tree $S(T)$ have

Table 2. Comparison with $R(v) = 2$: JOIN - proposed algorithm, BBT - Balanced Binary Tree

# hosts	JOIN			BBT		
	$S(T)$	$H(T)$	EDD	$S(T)$	$H(T)$	EDD
10	13	3	5	18	5	12
100	103	6	8	190	14	26
500	503	8	10	920	22	40
1000	1003	9	11	1890	44	78

the same value in the tree with $R(v) = 2$ and $R(v) = 3$. The reason is that the algorithm always chooses the nearest neighbour and the routing costs are kept minimal. The End-to-End Delay depends on the height of the tree and the $R(v)$ value respectively.

The chart 11 shows the End-to-End delay dependency graphically.

If the EED-value of the tree generated by the algorithm $JOIN$ increases only slightly, then the EED-value of the binary balanced tree generated increases dramatically. In the real $P2P$ overlay multicast tree each peer is able to serve different number of outgoing resources. The trend of EED delay dependency will remain definitely similar.

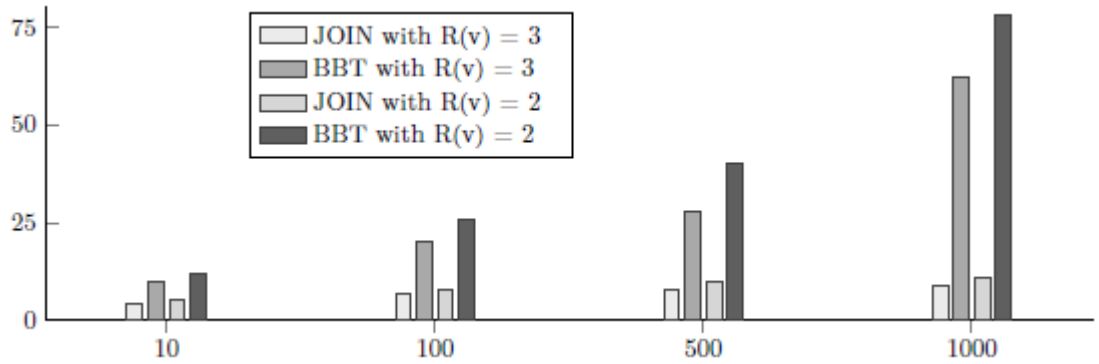

Figure 11. Comparison of results: JOIN - proposed algorithm, BBT - Balanced Binary Tree

6. CONCLUSION

In this paper we presented a novel multicast tree construction and maintenance approach based on the topological network coordinates of the end hosts. The algorithms presented in this paper were developed to achieve the following desirable properties:

– Minimal routing overhead in the underlying network
– Optimal resource management of the hosts
– Short end-to-end delay

We evaluated our approach theoretically and by using simulations. Compared to the randomly generated trees our approach improves significantly the performance metrics of a multicast overlay tree. Our future work will concentrate on implementing this approach in a real environment, collecting and analysing the performance data.

REFERENCES

[1] Abboud, O., Pussep, K., Kovacevic, A., Mohr, K., Kaune, S., Steinmetz, R., Enabling Resilient P2P Video Streaming, Multimedia Systems, Vol. 17, No. 3, p. 177-197, June 2011

[2] Jurca, D., Chakareski, J.,Wagner, J., Frossard, P., Enabling Adaptive Video Streaming in P2P Systems, IEEE Communications Magazine, p. 108-114, June 2007

[3] Tran, D. A., Hua, K., Do, T., ZIGZAG: An Effcient Peer-to-Peer Scheme for Media Streaming, Proc. of IEEE INFOCOM, Vol.2, pp.1283-1292, 2003

[4] Márk Jelasity and Ozalp Babaoglu, T-Man: Gossip-based overlay topology management, 3rd Int. Workshop on Engineering Self-Organising Applications (ESOA'05), Springer-Verlag, pp. 1-15, 2005

[5] X. Liao, H. Jin, Y. Liu, L. M. Ni, D. Deng., AnySee: Peer-to-peer live streaming. In Proceedings of IEEE International Conference on Computer Communications, Barcelona, Spain, 2006.

[6] Magharei, N., Rejaie, R., Yang G., Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches, 26th IEEE International Conference on Computer Communications-INFOCOM, pp. 1424-1432, 2007.

[7] H. Byun and M. Lee, HyPO: A Peer-to-Peer based Hybrid Overlay Structure, IEEE ICACT 2009, Feb. 2009.

[8] Awiphan, S., Zhou Su, Katto, J., Two-layer Mesh/Tree Overlay Structure for Live Video Streaming in P2P Networks, proc. 7th IEEE Consumer Communications and Networking Conference (CCNC), pp. 1-5, 2010

[9] F.,Wang, Y., Xiong, and J., Liu, mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming, IEEE Transactions on Parallel and Distributed Systems, Vol. 21, No. 3, pp. 379-392, March 2010.

[10] Donald Knuth, The Art of Computer Programming, Addison-Wesley, Vol. 3,1973

[11] Zhang, X.Y., Zhang, Q., Zhang, Z., Song, G., Zhu, W., A Construction of Locality-aware Overlay Network: mOverlay and its Performance, IEEE Journal on Selected Areas in Communications, pp. 18-28, 2004

[12] Banerjee, S., Bhattacharjee, B. Kommareddy, C., Scalable application layer multicast, Proc. ACM SIGCOMM Conf., ACM Press, New York, 2002.

[13] Abboud, O., Kovacevic, A., Graffi, K., Pussep, K., Steinmetz, R., Underlay Awareness in P2P Systems: Techniques and Challenges, IEEE International Parallel and Distributed Processing Symposium, 2009

[14] Xuping Tu, Hai Jin, Xiaofei Liao, and Jiannong Cao, Nearcast: A locality-aware P2P live streaming approach for distance education. ACM Transactions on Internet Technology, Vol. 8 - Issue 2, 2008

[15] T. S. Eugene Ng, Hui Zhang, Predicting internet network distance with coordinates-based approaches. Proc. of IEEE INFOCOM, New York, Vol. 1, pp. 170–179, 2001

[16] James A. Muir and Paul C. Van Oorschot, Internet geolocation: Evasion and counterevasion, Journal ACM Computing Surveys, Vol. 42 - Issue 1, No. 4, December 2009

[17] Editor: Andrei Popescu, Geolocation API Specification, W3C, 22 December 2008

[18] Editor: Philip Olson, PHP Manual - Geo IP Location, The PHP Documentation Group, 2014, http://php.net/manual/en/book.geoip.php

[19] Chao Dai, Yong Jiang, Shu-Tao Xia, Hai-Tao Zheng, and Laizhong Cui. A traffic localization strategy for peer-to-peer live streaming. In 2013 IEEE Symposium on Computers and Communications, ISCC 2013, Split, Croatia, 7-10 July, 2013, pages 495–501, 2013.

[20] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel F. Gryniewicz, and Yixin Jin. An architecture for a global internet host distance estimation service, Proceedings of IEEE INFOCOM, 1999.

[21] Ethan Katz-bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards IP Geolocation Using Delay and Topology Measurements, IMC, 2006