EVALUATION OF A TOPOLOGICAL DISTANCE ALGORITHM FOR CONSTRUCTION OF A P2P MULTICAST HYBRID OVERLAY TREE

Sergej Alekseev[1] and Jörg Schäfer[2]

[1]Department of Computer Science and Engineering, Computer Networks and OS,Frankfurt University of Applied Sciences, Germany
[2]Department of Computer Science and Engineering, Distributed Systems,Frankfurt University of Applied Sciences, Germany

ABSTRACT

In the last decade Peer to Peer technology has been thoroughly explored, becauseit 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 experimental 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 following 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 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.

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.

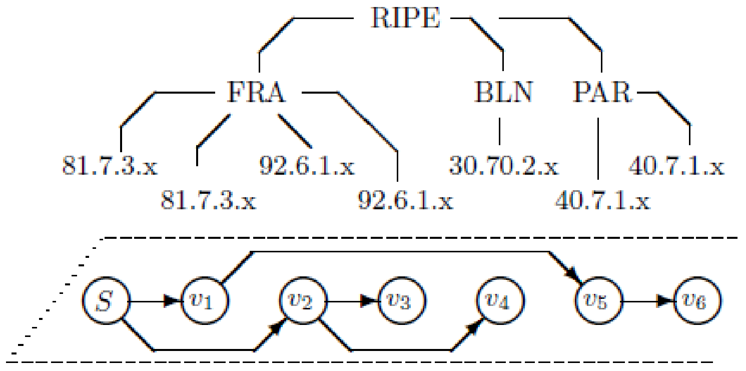


Figure 1.Topological search tree and p2p multicast tree structure.

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 Ts 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 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 Ts is considered as an H-layered topological tree. According to the topology of the search tree Ts for each vertex v ∈ V the network coordinate C(v) is defined as follows:

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

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 Ts can be defined using tuple notation as Ts = (Vs, Es), 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 Vxand Vy with the static network coordinates:

$$C(vx) = \{C0(vx), \ldots Ci(vx) \ldots CH-1(vx)\}$$
$$C(vy) = \{C0(vy), \ldots Ci(vy) \ldots CH-1(vy)\}$$

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

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

e.g. for the following vertices

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

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 C0, C1, . . . Ci, formally:

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

In the example above the *LCC*(*Vx*, *Vy*)=*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 Ts includes the coordinates C(S) (fig. 3a). To attach the new node V1 the joining algorithm extends the search tree Ts by the adding the coordinates C(v1) 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 v1 is attached to the host S (fig. 3b). The fig. 3c illustrates the attaching of the host v2 to the multicast overlay tree.

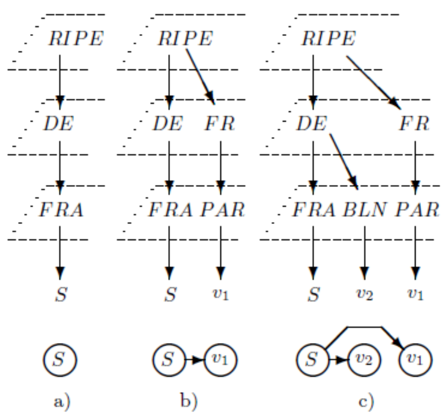


Figure 3. An example of the algorithm JOIN execution.

To show that the routing in the constructed multicast tree T is optimally organised, we assign to each edge e = {v0, v1} a topological distance value D(e) := D(v0, v1), that represents the routing costs between the vertices v0 and v1. 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) is defined as

$$S(T) := \sum_{e \in E} D(e) \tag{6}$$

and is a measure of the total routing costs in the tree in idealised units, see also section 5. The lower the value S(T) is, the less 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 we will show in section 6 this is confirmed by our experimental results.

Now 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 T1 with S(T1) < 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 Vn+1 be the next vertex added to the multicast tree T and V ∈ T is the nearest neighbour of Vn+1 (fig. 4a). Let us show that S(T) + D(V, Vn+1) is minimal.

Consider any multicast trees T^t, where v_n+1 not connected to v. We will see that there is no tree with $S(T^t) < S(T)$.

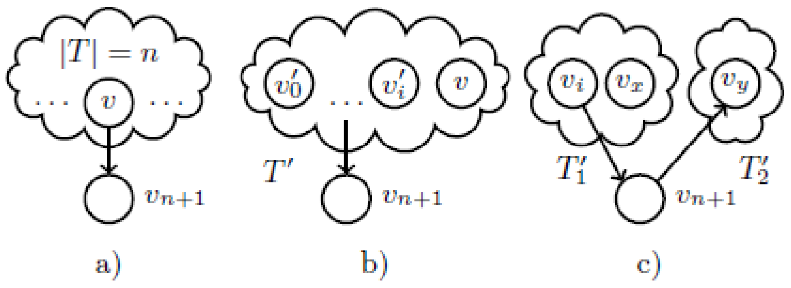


Figure 4. Proof by induction.

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:

$$S(T') < S(T) + D(v, v_{n+1}) \tag{7}$$

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 (7) we get:

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

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

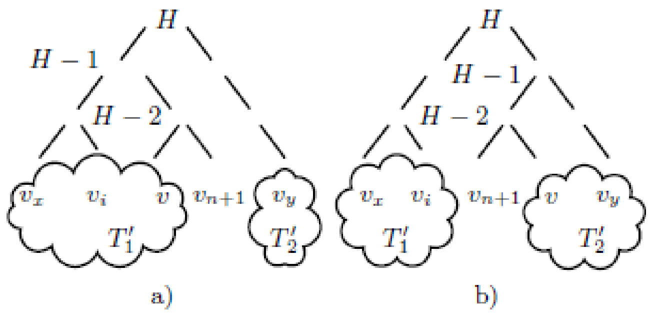


Figure 5. Topological distances.

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

$$\begin{aligned}(D(v_x, v_y) = D(v_{n+1}, v_y)) \wedge (D(v_x, v_y) > D(v_i, v_{n+1}))\\ (D(v_x, v_y) > D(v_{n+1}, v_y)) \wedge (D(v_x, v_y) = D(v_i, v_{n+1}))\end{aligned} \tag{8}$$

But (8) contradicts inequality (7). Thus S(T) + D(v; vn+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 T0 and T1 for the same set of vertices (end hosts) with *S(T0)* ≠S(T1).

According to theorem 1 the value of S(T0) and the value of S(T1) are minimal. Since S(T0) ≠ S(T1), it follows that either S(T0) or S(T1) 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*(υ), is a maximum number of outgoing links e € E, which can be served by the vertex v. The value R(υ) is calculated based on available bandwidth and other resources of the peer.

The pseudocode of the joining algorithm with the peer-resource management JOINR is shown in fig. 6 (we call υ in LCC(*new, n*) iff LCC*(new*; υ) = LCC(*new, n*)). To join a

```
Algorithm $JOIN_R$ ($new$, $T_s$)

$T_s = T_s \cup C(new)$
/* $n$ is a potential neighbour of $new$ */
for (all reachable hosts $v$ in $LCC(new, n)$) {
      if($R(v)$ != 0) {
         attach $new$ to $v$;
         $R(v) = R(v) - 1$;
         return;
      }
}
for ( all reachable hosts $v$ in $LCC(new, n)$ ) {
      for (all hosts $v_x$ connected to $v$) {
         if($D(v, new) \leq D(v, v_x)$) {
            insert $new$ between $v$ and $v_x$;
            $R(new) = R(new) - 1$;
            return;
         }
      }
}
```

Figure 6. The pseudocode of the JOINR algorithm

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

Figure 7 illustrates an example of the algorithm *JOINR* execution. We define for this

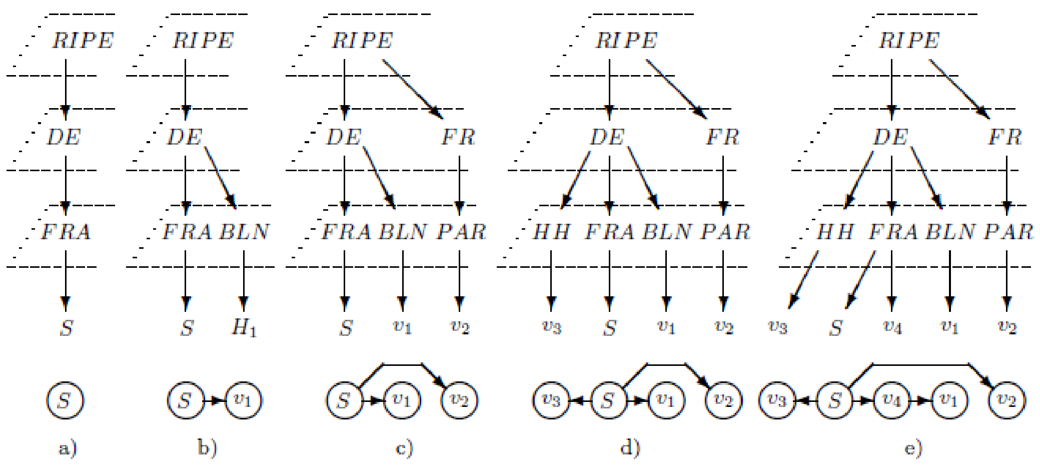


Figure 7. An example of the algorithm *JOINR* execution.

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 $v2$. After updating the search tree Ts the algorithm *JOINR* 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 Ts from the vertex $v2$ until the first branch. The potential nearest neighbours of $v2$ are the hosts S and $v1$, because D(S, $v2$) = *D*($v1$; $v2$) = 2. The host v2 is attached to the host S and the R(S) value is updated accordingly *R(S)* = 2 - 1 = 1.

The attaching of the host v3 (fig. 7d) is similar to the previous step. The R(S) value is pdated 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 $v4$. The nearest neighbour of $v4$ is S. But *R(S)* = 0 and there are no other free potential neighbours with the same topological distance. The algorithm *JOINR* checks in this case all potential nearest neighbours v whether any hosts vx with D(v, $v4$) ≤ 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(υ) that represents the maximum number of outgoing data links which can be served by the vertex. We call *R(T)* solved $\forall v \in V : R(v) \leq R_{max}(v)$. 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:

$$\forall v \in V : R_{max}(v) > 0 \tag{9}$$

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

Proof. The algorithm *JOINR* 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 9 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(υ, *x*) _ D(υ, *new*) (fig. 8a). The vertex *x* is reconnected to the vertex new (fig. 8b). D(υ, *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

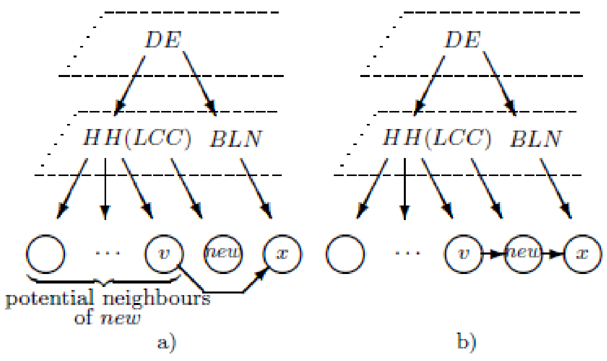


Figure 8. Proof of the second loop.

(theorem 1). So S(T) is minimal. According the precondition 9 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.

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 $p = \{v_0, \ldots v_n\}$ 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  $T_s = T_s \cup C(new)$
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($D(v, new) \leq D(v, v_x)$) {
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. EVALUATION METRICS

As a basis for performance evaluation, the following raw-data and statistics about p2p applications are generated by our platform:

- Signaling and connection state events as described in [26]
- Geolocation and the available bandwidth information about peers
- Topological network coordinates according the definition in [27]
- The p2p overlay topology as a graph
- Sent/received and lost packets by each peer
- Round trip time (RTT) between two peers
- Delay and jitter time of received packets

```
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

Based on the collected data we are calculating the following metrics (explained in section 5.1) to evaluate the performance and other network characteristics of the WebRTC sessions:

- Link Stress (LS), Average Link Stress (ALS), and Topological Link Stress (TLSj) for each layer j
- Resource Usage (RU) and Topological Distance Sum S(T)
- Peer Degree (PD) and Peer Stretch (PS)
- Latency and End-to-End Delay (EED)
- Absolute/Relative Delay Penalty (ARDP or RDP)
- Absolute Peer Join and Reconfiguration Delay
- Total Sent/Received Bytes
- Packet Retransmission

5.1. Link Stress, Resource Usage/Topological Distance Sum, Peer Degree and Peer Stretch

Link Stress, Average Link Stress, and Topological Link Stress

To evaluate the efficiency of multicast overlay trees, *Link Stress* LS_i has been defined in [22] and evaluated in [23] as the total number of identical copies of data packets over a physical link L_i. It accurately reflects the bandwidth efficiency of different approaches. We define the Average Link Stress as $ALS := 1/n \sum_{i=0}^{n} LS_i$, where LS_i denotes the Link Stress of an individual link.
Note, that although not explicitly stated in the definition, usually the units are chosen such that LS_i is measured *per message* or *per bytes* transferred in order to normalise this

measure appropriately. In the sequel, we shall always understood LS_i in the sense of copies per logical (payload) message – i.e. if e.g. $LS_i = 3$, a single payload message results in three (identical) messages send over the network over this link.
In our set-up it makes sense to further define the *Topological Link Stress* (TLS_j) on layer j as the average of the Link Stress on all links of layer j, i.e. as:

$$TLS_j := \frac{1}{|\{h(L_i) = j\}|} \sum_{h(L_i)=j} LS_i, \tag{11}$$

where $h(L_i)$ denotes the height of a Link L_i. It is clear from the definitions that the Average Link Stress (ALS) is just the weighted average of the Topological Link Stress (TLS_j), i.e.

$$ALS = \sum_{j=0}^{H} \alpha_j TLS_j, \tag{12}$$

where $\alpha_j := \frac{1}{n} |\{h(L_i) = j\}|$.
Note, that link stress might be caused not only due to sub-optimal topology of the *P2P* multicast tree which our algorithm tries to avoid, but also due to technology related problems like e.g. the need to retransmit messages.
In the literature, besides Link Stress also Nodes Stress is known, see e.g. [28]. However, as node stress is more interesting in a purely distributed set-up and our algorithm is a hybrid one, node stress is less relevant to us and therefore we do not calculate it here.

Resource Usage and Topological Distance Sum

In [22] *Resource Usage* is defined as

$$RU = \sum_{i=1}^{n} D_i * LS_i, \tag{13}$$

where n is the number of links active in data transmission, and D_i is the delay of link i. As argued in [22], "the resource usage is a metric of the network resources consumed in the process of data delivery to all receivers. Implicit here is the assumption that links with higher delay tend to be associated with higher cost."
In the context of topological distances it is interesting to compare RU with the *Topological Distance Sum* $S(T)$. From the definition (eq. 6) it is clear that $S(T) = \sum_{i=1}^{n} D(L_i)$. Henceforth, both measures are sums of contributions of individual link (edge) costs. Whereas the RU equates costs per link with delays per message necessary for transmission, $S(T)$ proxies costs by topological distances (in "idealised" uniform units). The experiments described below provide insight how much the two measures are different in a real scenario (and not just using different, but equivalent units).
In [22] the *Normalized Resource Usage* (NRU) was introduced as well. It is a relative measure comparing the RU with the (optimal) RU based on the standard DVMRP algorithm:

$$NRU := RU_{Actual}/RU_{DVRMP} \tag{14}$$

Peer Degree and Peer Stretch

Peer Degree (PD) is the number of incoming and outgoing links. Peer Stretch (PS) is the ratio between the length of the data path from the server to a peer in our multicast tree to the length of the shortest path between them in the underlying network.

5.2. Latency, End-To-End Delay And Absolute/Relative Delay Penalty

End-to-End Delay

In the literature, End-to-End Delay (EED) is the RTT between two neighbouring peers, see [14]. To avoid any confusion with the (topological) End-to-End Delay defined in eq. 10, we denote the (topological) End-to-End Delay by T-TED in the sequel, in particular in table 5 and figure 14.

Latency

Latency as defined in [23] is a metric measuring the end-to-end delay from the source to the receivers, as seen by the application. It includes the propagation and queuing delays of individual overlay links, as well as queueing delay and processing overhead at end systems along the path. We ideally wish to measure the latency of each individual data packet. However, issues associated with time synchronisation of hosts and clock skew adds noise to our measurements of one-way delay that is difficult to quantify. Therefore, we choose to estimate the round trip time (RTT). By RTT, we refer to the time it takes for a packet to move from the source to a recipient along a set of overlay links, and back to the source, using the same set of overlay links but in reverse order. Thus, the RTT of an overlay path S-A-R is the time taken to traverse S-A-R-A-S. The RTT measurements include all delays associated with one way latencies, and are ideally twice the end-to-end delay.

Relative Delay Penalty

Relative Delay Penalty (RDP) is defined as the ratio of the delay between the source and a receiver along the overlay tree to the unicast delay between them.

We do not use the RDP metric, because it is not always possible to determine the unicast delay value. Instead the Absolute Delay Penalty (ADP) is used. ADP is defined as the latency for propagating a data packet along the overlay path from the source host to an destination host, i.e. $ADP := \sum_{i=1}^{n} D_i$, where, n is the number of links along the overlay path and D_i is the delay of link i.

Another common metric is the *Average Relative Delay Penalty* ($ARDP$) as defined in [24]. $ARDP$ is the average RDP between all node pairs, i.e. $ARDP = \frac{1}{N(N-1)} \sum_{\substack{i,j=0 \\ j \neq i}}^{N} D'_{i,j}/D_{i,j}$, where N is the number of peers in the overlay and $D'_{i,j}$ and $D_{i,j}$ denote the latency of a node pair i and j in the overlay and the physical network resp. Smaller ARDP is an indication that most node-pair latencies on the overlay are close to latencies on the physical network, see [25] for details. As we cannot directly compute the delay of the underlying physical network in our framework, we do not compute this metric.

5.3. Total Sent/Received Bytes And Packet Retransmission

These metrics just count the total number of bytes for send and received bytes and the total number of packet retransmissions.

6. Experimental Evaluations

In this section, we report and discuss the results for the performance evaluation of the proposed approach using a topological tree (TT), in comparison with the Balanced Binary Tree (BBT) scheme. We choose BBT for comparison because it has the optimal broadcasting time of logarithmic order and many approaches and applications are based on this scheme.

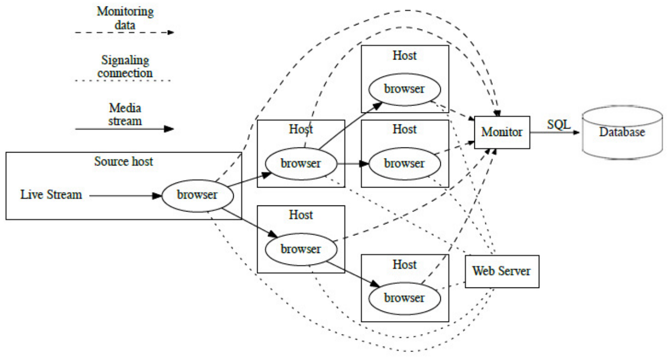


Figure 11. The platform architecture

We have implemented aWebRTC [29] based platform for testing and monitoring p2p broadcasting algorithms. The platform provides an WebRTC based interface for implementing p2p algorithms and a monitoring interface to collect performance information. Figure 11 shows the system architecture of our platform deployment. Referring to this figure, each end system contains of a device with a WebRTC capable browser. The p2p algorithm is running on the central server. The end hosts initialise the join process by loading a web page with a video element and submits the topological coordinate, calculated via HTML5 Geolocation Browser API [30], to the central data base. The browser sends a JOIN-request to the central server, establishes a peer connection and relays streams. Each host sends the performance statistics to the monitor asynchronously. The monitor is responsible for logging the performance information of the joined hosts as mentioned in section 5.

We executed many experiment with our cooperation partner on Vietnamese-German University (VGU), which covers two continents, several subnetworks in the different cities in Germany and Vietnam. Each experiment consists of the broadcasting a video stream and a randomly organized joining process. The duration of each experiment is three minutes. The number of hosts was increased from 10 to 100 hosts.

From the all experiments, we present a selected experiment of a small size in detail, which includes 7 hosts from Germany and 8 hosts from Vietnam. The collected and calculated statistics of this experiment are transferable to the experiments with the larger number of hosts and the derived trends are very similar. The figure 12 represents the balanced binary tree and the figure 13 the topological broadcast tree with the corresponding search tree according our aproach. The collected performance statistics for balanced binary and topological broadcast tree are listed in the tables 1 and 2 accordantly.

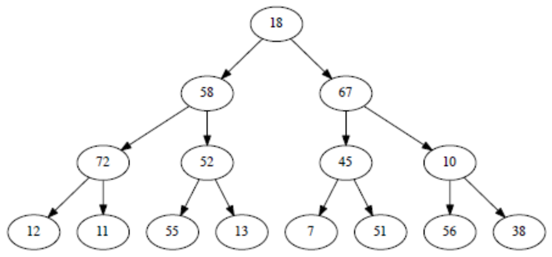


Figure 12. The balanced binary tree with 15 nodes

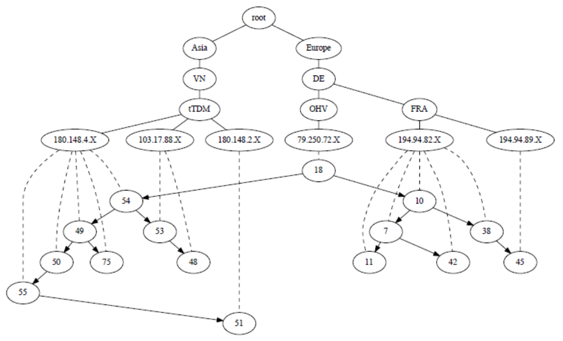


Figure 13. The topological tree with 15 nodes and the corresponding search tree

6.1. METRICS

Link Stress

We cannot directly measure the link stress as it is impossible to capture the data at the source. However the link stress (per message, compare remark in section 5.1) can be estimated by calculating the ratio of all packets over the max number of packets going over a link of a certain level – ignoring the (negligible) number of lost packets, see table 2. The result is depicted in table 3. As one can see from the figures, the link stress in the case of the balanced binary tree is particularly high for the level 6 and 4 links corresponding to continent and national links, namely 6.22 and 1.94. Our on topological distance based algorithm measures show a link stress of one by construction. Henceforth, also the average link stress ALS is much higher (4.12) than in the optimal case, see table 4. This corresponds to the S(T) which is 29 Vs 67 and is indeed smaller for our algorithm as it should be.

Table 1. Balanced Binary Tree Statistics

# link	*rtt in ms*	packets sent	packets lost	$H(T)$	Description
18 -> 58	270.857	66530	0	4	NATIONAL-LINK (OHV -> FRA)
18 -> 67	278.464	62584	0	4	NATIONAL-LINK (OHV -> FRA)
58 -> 72	1.185	60025	0	0	LOCAL-LINK (194,94,82,X -> 194,94,82,X)
58 -> 52	1.333	62058	0	0	LOCAL-LINK (194,94,82,X -> 194,94,82,X)
67 -> 45	21.333	59534	2	2	SUBNET-2-LINK (194,94,82,X -> 194,94,89,X)
67 -> 10	361.320	51463	159	6	CONTINENT-LINK (Europe -> Asia)
72 -> 12	370.250	41150	0	6	CONTINENT-LINK (Europe -> Asia)
72 -> 11	380.684	41003	101	6	CONTINENT-LINK (Europe -> Asia)
52 -> 55	358.315	9035	112	6	CONTINENT-LINK (Europe -> Asia)
52 -> 13	647.235	38152	154	6	CONTINENT-LINK (Europe -> Asia)
45 -> 7	348.204	38025	132	6	CONTINENT-LINK (Europe -> Asia)
45 -> 51	545.115	39102	184	6	CONTINENT-LINK (Europe -> Asia)
10 -> 56	2.215	32098	0	3	SUBNET-1-LINK (180.148.4.X -> 103.17.88.X)
10 -> 38	221.583	31987	78	6	CONTINENT-LINK (Asia -> Europe)

Table 2. Topological Tree Statistics

# link	*rtt in ms*	packets sent	packets lost	$H(T)$	Description
18 -> 54	391.741	54254	1	6	CONTINENT-LINK (Europe -> Asia)
18 -> 10	250.307	52947	274	4	NATIONAL-LINK (OHV -> FRA)
54 -> 53	2.112	53806	0	3	SUBNET-1-LINK (180.148.4.X -> 103.17.88.X)
54 -> 49	3.385	47246	0	0	LOCAL-LINK (180.148.4.X -> 180.148.4.X)
55 -> 51	1.427	42356	0	0	LOCAL-LINK (180.148.2.X -> 180.148.2.X)
10 -> 7	1.339	52879	0	0	LOCAL-LINK (194.94.82.X -> 194.94.82.X)
10 -> 38	5.571	48357	0	0	LOCAL-LINK (194.94.82.X -> 194.94.82.X)
7 -> 42	1.260	50110	0	0	LOCAL-LINK (194.94.82.X -> 194.94.82.X)
7 -> 11	23.521	50382	0	0	LOCAL-LINK (194.94.82.X -> 194.94.82.X)
38 -> 45	18.287	46454	0	2	SUBNET-2-LINK (194,94,82,X -> 194,94,89,X)
53 -> 48	2.431	47354	0	0	LOCAL-LINK (103.17.88.X -> 103.17.88.X)
50 -> 55	2.289	43233	0	1	SUBNET-3-LINK (180.148.4.X -> 180.148.2.X)
49 -> 50	1.763	44843	0	0	LOCAL-LINK (180.148.4.X -> 180.148.4.X)
49 -> 75	1.248	43296	0	0	LOCAL-LINK (180.148.4.X -> 180.148.4.X)

Table 3. Link Stress

	Link Level	6	4	3	2	1	0
Algorithm	TT	1.00	1.00	1.00	1.00	1.00	1.00
	BBT	6.22	1.94	1.00	1.00	NA	1.00

Resource Usage

As a consequence, the resource usage RU of our algorithm is much smaller than the resource usage of the Balanced Binary Tree Algorithm, see table 4. The Normalized Resource Usage of our algorithm is close to the optimal one, whereas the Normalized Resource Usage of the Balanced Binary Tree Algorithm exceeds the one of the DVMRP by a factor of 30, see as well table 4. If one compares S(T) and RU, one deducts that RU magnifies the lesser efficiency of BBT vs. TT compared to S(T). This is seemingly due to the effect of the higher link stress amplifying the delay penalties of the intercontinental links.

Table 4. Aggregated Performance Measures

		S(T)	ALS	RU	RU(DVMRP)	NRU	No. of Messages
Algorithm	TT	29	1.00	353.34	343.45	1.03	677517
	BBT	67	4.12	10594.04	350.30	30.24	662746

Peer Degree and Peer Stretch

Peer Degree (PD) of both algorithms is mostly identical. The balanced binary tree algorithm has by definition one incoming and a maximum two outgoing links. In the topological algorithm we defined for all nodes the R(v) = 2 (maximum number of outgoing links) for the better comparison. However, the topological tree has a node of just one outgoing link. The Peer Stretch (PS) of the Balanced Binary Tree in comparison to our algorithm has the lower value: PS(BBT) = 3, whereas PS(TPT) = 5. The balanced binary tree has always the optimal peer stretch by definition. But the multicast tree should not have small stretch to keep the end-to-end delay short. On the contrary, our measurements show that this is counterproductive.

Delays

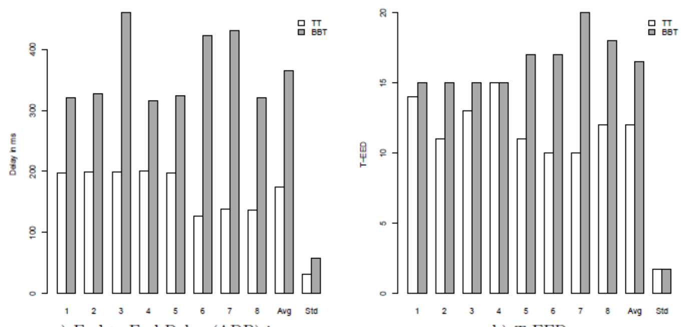
The end to end delays (ADPs) are depicted in table 5 and figure 14 for 8 destination nodes. As one can see from the numbers, the delays for our algorithm are less than half of that of the Balanced Binary Tree Algorithm. The delays are also more predictable as the standard deviation shows[4]. The T-EED is less for our algorithm compared to BBT.

Table 5. Delays in Milliseconds and T-EED

Dest. Node	1	2	3	4	5	6	7	8	Total	Average	STD
ADP (TT)	198.14	198.44	199.59	200.30	198.19	126.45	137.58	137.08	1395.78	174.47	31.74
ADP (BBT)	321.15	326.36	459.71	315.25	324	422.46	430.68	321	2920.61	365.08	57.11
T-EED (TT)	14	11	13	15	11	10	10	12	96	12	1.70
T-EED (BBT)	15	15	15	15	17	17	20	18	132	16.50	1.70

Messages Sent

As depicted in table 4 we send roughly the same amount of messages (677517 and 662746 messages resp.) in both scenarios.



a) End-to-End Delay (ADP) in ms b) T-EED

Figure 14. End-to-End Delays and T-EED for Destination Nodes

7. 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 execution several experiments in the real network. Compared to the balanced binary trees our approach improves significantly the performance metrics of a multicast overlay tree.

Our future work will concentrate on collecting, analysing further performance data in a real environment and improving our algorithms.

ACKNOWLEDGEMENTS

The authors would like to thank Nhu Tran Quang from Vietnamese-German University for supporting our experiments and Johannes Kinzig from Frankfurt University of Applied Sciences for the support in preparation of statistics.

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, ISCC2013, 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

[22] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, A case for end system multicast, in Proc. of ACM SIGMETRICS, 2000

[23] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, Enabling conferencing applications on the Internet using an overlay multicast architecture, In Proceedings of ACM SIGCOMM, p55-67, 2001

[24] W. Wang, D. Helder, S. Jamin, and L. Zhang, Overlay Optimizations for End-host Multicast, Networked Group Communications, 2002

[25] Wenjie Wang and Cheng Jin and Sugih Jamin, Network Overlay Construction under Limited End-to-End Addressability, In Proc. of IEEE INFOCOM 05, 2004

[26] S. Alekseev, C. von Harscher and M. Schindler, Finite State Machine based Flow Analysis for WebRTC Aplications, Fourth International Conference on Innovative Computing Technology (IEEE INTECH), University of Bedfordshire, Luton, UK, 2014

[27] S. Alekseev, J. Schäfer, A New Algorithm for Construction of a P2P Multicast Hybrid Overlay Tree Based on Topological Distances, The Seventh International Conference on Networks & Communications (NeCoM 2015), Zürich, Switzerland, 2016

[28] Miguel Castro, Michael B. Jones, Anne-Marie Kermarrec, Antony Rowstron, Marvin Theimer, Helen Wang and AlecWolman, An Evaluation of Scalable Application-Level Multicast Built Using Peer-to-Peer Overlays, Infocom'03, 2003

[29] Adam Bergkvist, Daniel C. Burnett, Cullen Jennings, Anant Narayanan, WebRTC 1.0: Real-time Communication Between Browsers, W3C Working Draft 10 February 2015 https://www.w3.org/TR/webrtc/

[30] Andrei Popescu, Google, Inc, Geolocation API Specification, W3C Editors Draft 11 July 2014 http://dev.w3.org/geo/api/spec-source.html