Center Placement Algorithms for Large Multicast Groups

Alexander S. Weigmann and Jörg Nonnenmacher and Ernst W. Biersack*
Institut EURECOM, 2229 route des Crêtes
B.P. 193, 06904, Sophia Antipolis Cedex, FRANCE
Tel/Fax: +33 493002626/+33 493002627
{weigmann, nonnen, erbi}@eurecom.fr

October 1996

Abstract

An increasing number of distributed applications require a specific form of multicast called dissemination, in which a single source *reliably* transfers data to multiple receivers. Reliability requires that data packets are acknowledged positively or negatively, which leads for large groups of receivers (100s or 1000s of participants) to the problem of feedback implosion. Among the approaches trying to avoid feedback implosion, the cluster approach is the most promising. It partitions the multicast delivery tree into clusters. Each cluster has a representative called center, which is used for acknowledgment accumulation and local retransmission. Up to now, clustering/center placement has been done administratively or based on network addresses. Needed are center placement algorithms, allowing the introduction of placement criteria based on the network topology and on delay. In this work, three center placement algorithms designed for static multicast groups are presented and simulation results are shown in order to asses their performance.

1 Introduction

Emerging high speed networks and the widespread availability of multimedia capable workstations have drastically increased the demand for distributed applications. Along going with this evolution comes an increasing need for multicast, allowing to distribute data to many users while efficiently using scarce communication bandwidth.

Many applications require a specific form of multicast called dissemination [1, 19, 2, 20], in which a single source *reliably* transfers data to multiple receivers. Applications include electronic newspaper distribution, WWW in-advance caching, file transfers and shared whiteboard (see also [1, 13]).

While some applications, like the shared whiteboard, support *dynamic* multicast groups, where receivers dynamically join or leave the group during transmission time, numerous other ones, like newspaper distribution or file transfers, are designed for *static* groups of receivers, which remain unchanged throughout the entire transmission.

IP Multicast [4], which forms the basis of the MBone [6, 16] provides scalable, but not reliable multicast. Reliability requires that either data packets are acknowledged by the receivers (ACKs) or that packet losses are reported using negative acknowledgments (NACKs). For larger groups of

*Please send all correspondence to E. Biersack receivers (100s or 1000s of participants), this leads to the problem of packet implosion at the source [15]: The source suffers from the significant overhead incurred due to the processing of the ACKs or NACKs from the receivers.

The research community proposed two directions to deal with the scalability problem:

- *timer approaches* [8, 7] that introduce a certain asynchronity among receivers to protect the source from an implosion and
- *cluster approaches* [20, 13, 9] that achieve the same goal by introducing a hierarchy into the multicast tree.

In the cluster based approaches, the multicast delivery tree is portioned into **clusters**. Every cluster has a representative, the **center**. Receivers do not acknowledge received packets directly to the source but only to the center of their cluster.

All centers are organized in a hierarchy. A center is responsible for all receivers in its cluster as well as for other centers located one level below in the cluster hierarchy. A center receives ACK packets from receivers and centers it is responsible for and forwards a single acknowledgment packet to the center in the next higher hierarchy level (ACK accumulation).

Most of the cluster based approaches presented so far either use centers placed administratively in the network [13] or rely on centers placed algorithmically *within* a group (cluster), which is either defined by hand or based on network– (subnet–) addressing [20, 11]. Administratively placed centers or clusters do not scale well for big groups of receivers and clustering based on addressing does not reflect the real network structure (connectivity and delay relations) but only the address structure.

Needed are center placement algorithms, allowing the introduction of placement criteria based on the network topology and on delay relations and not only on addressing schemes. In this work, criteria for a "good" center placement are stated. Three center placement algorithms designed for static multicast groups are presented and simulation results are shown in order to asses their performance.

The remainder of this paper is organized as follows. In section 2, the center placement problem is described in a formal way. In section 3, placement algorithms are presented and section 4 shows simulation results. In section 5, final conclusions are drawn.

2 The Center Placement Problem

The problem of placing centers in a multicast tree can be looked at from a graph-theoretical point of view. This allows to approach the problem without the need to consider implementational aspects, which are linked very closely to a specific type of network. Once the center placement problem is solved in theory it can be transfered to reality and implemented.

The graph-theoretical point of view abstracts from the real connection endpoints (receivers) and considers just multicast routers. Regarded is a directed **tree** (routers = vertices, links = edges) rooted at the multicast source. Edge weights represent link delays.

Receivers can be located on every vertex except on the tree's root. All leaves of the tree are receivers, but vertices that are not leaves can also be receivers.

Centers can be placed on all internal vertices, thus not on leaves. The root is implicitly a center. The **number of centers** in the tree is called $\boldsymbol{NOC}$.

The criteria for center location are based on a given **optimal cluster size** $\boldsymbol{Z}$ and the minimization of the end–to–end delay between the source S and the receivers $\in R$ in a multicast tree T.

The following definitions help to specify the criteria.

Definition 2.1 Directly Served
A receiver r_k (a center c_j) is **directly served** by a center c_i when no other center is located on the path in the tree T between c_i and r_k (between c_i and c_j). c_i is not directly served by itself.

Definition 2.2 Cluster with Center c_i
The **cluster with center c_i** is defined as the set of all the receivers and centers directly served by c_i, c_i itself, and all the edges interconnecting the elements of this set.

Definition 2.3 Cluster Size $cs(c_i)$
The **cluster size $cs(c_i)$** of the cluster with center c_i is the number of centers and receivers directly served by c_i.

With these definitions, the two criteria for center placement can be defined.

2.1 Cluster size

The cluster size plays a major role in the success of reliable multicast with clustering. Packet responses (ACKs or NACKs) require processing time and memory space at centers c_i that grow with the number of receivers or further centers directly served by c_i.

In order to limit processing time and memory size, a **cluster size constraint (CSC)** will be introduced, limiting the maximal cluster size to the optimal cluster size Z.

Definition 2.4 Cluster Size Constraint CSC
The **cluster size constraint CSC** requests that the size of any cluster with center $c_i \in C$ remains below or equal the optimal cluster size Z:

$$\forall c_i \in C : cs(c_i) \leq Z \tag{1}$$

As a measure for the deviation of the size of a cluster from the optimal value Z, a **squared error of the cluster size** is introduced:

Definition 2.5 Squared Error of the Cluster Size $SE(c_i)$
The **squared error of the cluster size $cs(c_i)$** of the cluster with center c_i is defined as the squared difference of Z and $cs(c_i)$:

$$SE(c_i) = (Z - cs(c_i))^2 \tag{2}$$

Two further terms can be derived from this definition: The **Accumulated Squared Error *ASE*** and the **Mean Squared Error *MSE*** of the cluster sizes. The latter one is the average squared error of the cluster size of all clusters in a tree.

Definition 2.6 Accumulated Squared Error of Cluster Sizes ASE
The **accumulated squared error of cluster sizes *ASE*** is defined as the sum of all squared errors of the cluster size $SE(c_i)$ in a tree.

$$ASE = \sum_{i=1}^{NOC} SE(c_i) = \sum_{i=1}^{NOC} (Z - cs(c_i))^2 \tag{3}$$

Definition 2.7 Mean Squared Error of Cluster Sizes MSE
The **mean squared error of cluster sizes $\boldsymbol{MSE}$** is defined as the Accumulated Squared Error of cluster sizes ASE divided by the total number of clusters in the tree.

$$MSE = \frac{ASE}{NOC} = \frac{1}{NOC} \sum_{i=1}^{NOC} (Z - cs(c_i))^2 \tag{4}$$

The MSE is taken as the first criterion for center placement. In future it will also be referred to as the *Cluster Size Criterion.*

2.2 End–to–End Delay

In multicast connections the end–to–end delay has a big influence on the performance of a reliable transport protocol. At every center c_i ACK-packets (acknowledge-packets) from directly served centers and receivers must be processed on the transport level. The end–to–end delay between a receiver r_j and the source S is composed of the delay of the links the packet transverses and a number of **center delays $\boldsymbol{cd}$** experienced at every center. The cumulative delay is defined as the **receiver delay $\boldsymbol{D(r_j)}$**.

Definition 2.8 Receiver Delay $D(r_j)$
The **receiver delay $\boldsymbol{D(r_j)}$** is the sum of the edge delays $d(v_k, v_l)$ on the path from the source S to a receiver r_j in the tree T plus the product of the number of centers on the path and a constant delay cd experienced at every center.

The end–to–end delay of the whole multicast connection is determined by the **maximal receiver delay $\boldsymbol{mDR}$**.

Definition 2.9 Maximal Receiver Delay mDR
The **maximal receiver delay $\boldsymbol{mDR}$** in a tree T is defined as

$$mDR = \max_{j=1..r} \{D(r_j)\} \tag{5}$$

The objective, referred to as the *Delay Criterion*, is to minimize the delay mDR.

Please note, that the objective is not the same as to minimize the maximal number of centers on the path from the source to a receiver. Having an arbitrary center placement, the path to the receiver that yields mDR may be different from the path where the most centers can be found on.

3 Center Placement Algorithms

Static multicast trees occur to be important in the context of reliable multicast as there exist numerous applications like newspaper distribution, WWW in–advance–caching and software distribution. All three examples have in common that the group of receivers is determined preceding the multicast transmission and that no further receivers will join or leave this group after the transmission has started.

Static multicast trees are also the easier case, as the center placement can be done in advance. The placement is done before the actual multicast transmission starts and is not time critical. Also extensive exchange of status information (for example information on the tree structure) between nodes would be possible (although not desirable).

In the following sections, three placement algorithms are presented:

- Center Placement by Split and Shift (CPSS)
- Center Placement by Shift and Merge (CPSM)
- Center Placement by Dynamic Programming (CPDP)

Two cases will be considered for every algorithm:

1. **Placement with cluster size constraint (CSC):** Center placement is done with respecting the cluster size constraint (CSC) defined in section 2.1. The CSC limits the number of directly served receivers and centers in a cluster to the optimal cluster size Z, thereby limiting status information that has to be present at the center.

   Note that if a center is placed on a vertex with an outdegree[1] greater than Z, the cluster size constraint cannot be met as the smallest feasible cluster size equals the outdegree.[2] This case is only a theoretical one. In practice, and also in the trees used for simulations, the maximal outdegree remains below the optimal cluster size Z.[3]

2. **Placement without cluster size constraint (CSC):** The cluster size constraint CSC is dropped. This allows more flexibility for center placement at the expense of a non-deterministic maximal cluster size. In practice this means that more memory must be made available at the nodes in order to store status information. Note that only the cluster size constraint is removed – the objective to minimize the mean squared error of the cluster size remains valid.

Before moving on to the different placement algorithms, two more expressions are defined. Although the algorithms can also be implemented without them, they allow to increase the efficiency in terms of execution speed of the algorithms both in the simulation environment and in a real implementation.

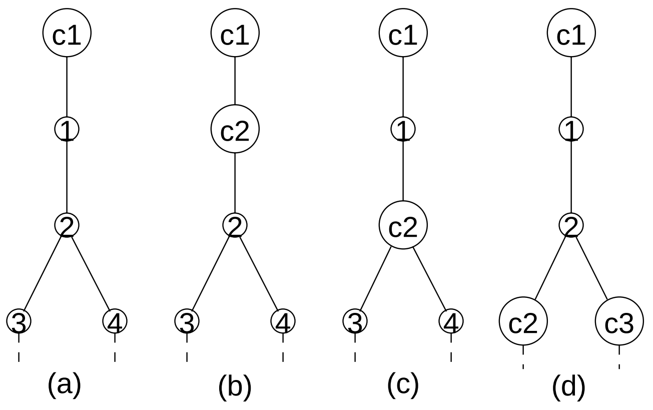


Figure 1: Furthest Directly Connected Descendant (FDCD). (a) Original tree, all nodes are routers (no receivers), the root is implicitly a center, (b) center on node 1: $cs = 1$, (c) center on node 2: $cs = 1$, (d) center on nodes 3 and 4: $cs = 2$.

Assume the following situation (see also Figure 1): A center c_i is located on a vertex v_i. Centers directly served by c_i are located on descendants of v_i. Looking at the size $cs(c_i)$ of the cluster with center c_i it can be noticed that the size of the cluster changes, if a directly served center is shifted

[1]The number of children of a node x in a tree T is called the *outdegree* of x.

[2]All children of the vertex the center is placed on are either leaves or internal vertices with centers.

[3]Work done by the Routing community considers average outdegrees between 2 and 8 [18] and maximal outdegrees of 4 and 5 [12] for networks. The outdegree for multicast trees is typically smaller than the outdegree in the network. The optimal value Z is not known yet; it will be in the order of 20–30.

- from a node with an outdegree > 1; or
- from a node that is a receiver

to its children. In order to formalize this effect, **directly connected** nodes and **furthest directly connected descendants (FDCDs)** are defined:

Definition 3.1 Directly Connected
A vertex $v_j \in V_t$ located in a tree $T = (V_t, E_t)$ is defined as being **directly connected** to an ancestor v_i if there are no other vertices with an outdegree > 1, centers or receivers on the path between v_i and v_j.

Definition 3.2 Furthest Directly Connected Descendant (FDCD)
In a tree $T = (V_t, E_t)$ a directly connected descendant $v_j \in V_t$ of a vertex $v_i \in V_t$ is called a **furthest directly connected descendant (FDCD)** of v_i if there exists no other directly connected vertex $v_k, k \neq j$ with a path between v_i and v_k that includes the path from v_i to v_j.

3.1 Split and Shift (CPSS)

The first algorithm called Center Placement by Split and Shift (CPSS) belongs to the group of *greedy* algorithms [3]. *Greedy* algorithms always make the choice that looks best at the moment. That is, they make a locally optimal choice in the hope that this choice will lead to a globally optimal solution.

CPSS starts the center placement by building the first cluster with its center located on the root of the tree. It places directly served centers on the root's furthest directly connected descendants (FDCDs) and subsequently tries to *shift* them towards the leaves. Centers are always shifted to their furthest directly connected descendants (FDCDs). Centers that must be shifted and that are located on a node with an outdegree > 1 are *split* and all FDCDs of this node become centers.

Once the cluster is created, the algorithm is repeated recursively for all placed (directly served) centers.

The directly served centers are shifted due to the following rules:

- Minimize the maximal receiver delay mDR.
- If the delay cannot be optimized any further, try to optimize the size of the cluster.

Minimizing the maximal receiver delay mDR does not imply that the total number of centers in the tree has to be as small as possible. It means that the number of centers on the longest path (in terms of delay) has to be as small as possible, as every center on the path to a receiver causes an additional center delay cd. The longest path *before* and the longest path *after* the center placement do not necessarily have to correspond.

In order to keep the number of centers on a certain path as small as possible, the centers on this path must be located as far apart from each other as possible. Following this rule, CPSS always tries to shift the center on the longest path towards the leaves. If a center is shifted onto a leaf, it will be removed since centers located on leaves do not make sense (they cannot serve any other node).

CPSS does not shift the center on the longest path if the cluster size constraint (CSC) is violated. In this case, CPSS tries to shift the center on the second longest path. If this one cannot be shifted either, it tries to shift the one on the third longest path and so on. If it does not succeed in shifting any center, the cluster is "finished" and the clusters for the just placed directly served centers can be built.

3.2 Shift and Merge (CPSM)

The second algorithm presented, called Center Placement by Shift and Merge (CPSM), also belongs to the group of *greedy* algorithms [3]. Unlike the CPSS algorithm, CPSM starts building clusters at the leaves and then moves subsequently towards the source. It concentrates on optimizing the cluster sizes (and thereby the mean squared error of the cluster size MSE). Less effort is spent in minimizing the maximal receiver delay mDR since CPSM does not explicitly try to reduce the number of centers on the longest path (in terms of delay), like CPSS does.

The first cluster is built by randomly choosing a leaf in the multicast tree and placing a center on this node (Figure 2 (a) and (b), node 4).

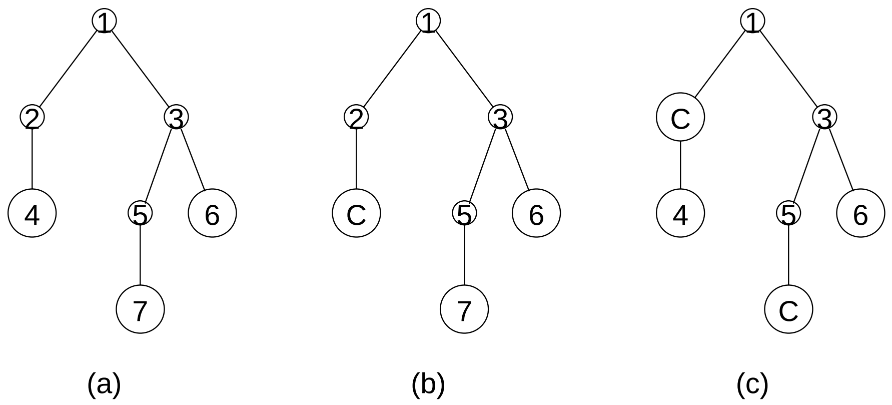


Figure 2: Center Placement by Shift and Merge (CPSM), part 1: (a) Tree without centers. Big circles represent receivers, small circles represent routers, (b) placing center on leaf, (c) shifting center towards root and placing new center on leaf in sibling's subtree.

In the following steps, this center will be *shifted* towards the root until it is located on a vertex that has one or more siblings[4] (node 2 in the example). Now, in every subtree rooted at one of these siblings, a center has to be placed on a randomly chosen leaf (Figure 2 (c), node 7), which is then again propagated towards the root. If a center encounters a node with one or more siblings (node 5), again, there has to be placed a center on a randomly chosen leaf (node 6) in every subtree rooted at a sibling and so on (Figure 3 (a)).

This leads to a situation, where a vertex (node 3) has two or more children a center is located on. Now a selection process starts, deciding, which of the centers are shifted to the parent node. First, this decision is made for the center on the longest path (in terms of delay), then for the center on the second longest path, then for the one on the third longest path etc. Note, that already placed centers contribute to the length of a path. If more than one center is shifted, the centers will be *merged* to a single center (Figure 3 (b)). If no center is shifted, a new center will be placed on the parent (Figure 3 (c), node 1).

The rules for deciding, which centers are moved, are the following:

- A center located on a leaf is always moved.
- If the new cluster size cs_{new}(parent (V)) *after* shifting a center on node V to its parent is closer to the optimal value Z than the cluster size cs (V) of V *before* the shifting and not higher than Z, the center will be moved.

[4]Two nodes having the same parent are called **siblings**.

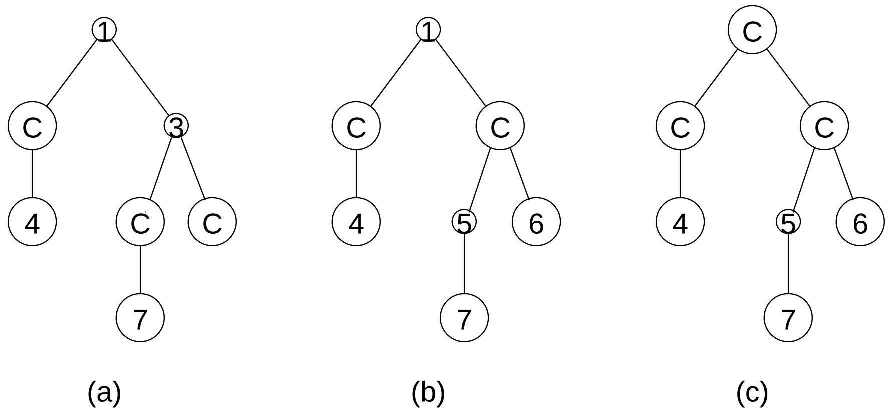


Figure 3: Center Placement by Shift and Merge (CPSM), part 2: (a) Shifting center towards root and placing new center on leaf in sibling's subtree, (b) two centers have been shifted to their parent and have been merged, (c) no center has been shifted, a new center will be placed on the parent.

- If the cluster size cs_{new}(parent (V)) *after* shifting a center on node V to its parent is equally close to the optimal value Z as the current cluster size cs (V) (without shifting) *and* if V lies on the path from parent (V) to the node in the subtree of parent (V) yielding the mDR *or* no center has been shifted to the parent of V yet, the center will be shifted.
- In all other cases, the center will not be shifted.

3.3 Dynamic Programming (CPDP)

The third algorithm presented is based on the method of dynamic programming, which is related to *divide-and-conquer* [3] and *branch-and-bound* [14] in the sense that it partitions a problem into subproblems and *intelligently* enumerates the feasible points of a combinatorial optimization problem.

To solve the problem, thus to find the best placement for a tree, dynamic programming is used in combination with a *local search*. Supposed, the best placements for *all* subtrees, as well as all important parameters of these placements, like the accumulated squared error in each subtree, the number of clusters in each subtree, the mDR in each subtree, etc. are known. Then, the cluster with its center located on the tree's root is built simply by placing directly served centers systematically on the root's descendants and by evaluating the different combinations. In practice, this means: Centers are placed on the root's furthest directly connected descendants (FDCDs) and then propagated towards the leaves. At every node with outdegree > 1, centers are split. Evaluation of the different center placements is done using a metric composed of the relative increase of the maximal receiver delay mDR and the normalized mean squared error of the cluster sizes (MSE):

$$Metric = 0.5 \cdot \frac{mDR_{with\ centers} - mDR_{without\ centers}}{mDR_{without\ centers}} + 0.5 \cdot \frac{MSE}{Z^2} \qquad (6)$$

Once, the optimal placement for a (sub–)tree has been found, the center configuration as well as the parameters characterizing the placement, like the accumulated squared error, the number of clusters, the mDR, etc. will be stored allowing them to be reused as the result of a solved problem.

In order to find the best placement, all feasible combinations must be evaluated and the number of feasible combinations is growing exponentially with the number of nodes in the tree. In order to limit the search to a bearable number of combinations, a **maximal search depth without improvement** ***MSD*** is introduced. If a certain center has been propagated for MSD steps without experiencing any improvement in the quality of the placement in terms of cluster size and delay, this center will be abandoned and not propagated any further. The introduction of this limit restricts the number of evaluated combinations, thereby limiting computational complexity, but also excluding numerous combinations. The optimal solution might not be found, the result has sub-optimal character.

4 Results

In the last sections, three center placement algorithms have been presented. In order to evaluate their performance, they were implemented and executed on a number of networks.

As input data, 10 networks with 200 nodes, an average outdegree of 3.0 and an average link delay of 412.38 were created following Doar [5], who modified the method of Waxman [17] to avoid an increasing outdegree for an increasing number of nodes. The network construction of Waxman/Doar is commonly used by the Multicast Routing community for comparing the performance of different multicast routing algorithms (see also Appendix A).

For each of the 10 networks, 10 different trees with 10, 20, 40, 60, 80, 100, 120 and 140 receivers each on randomly chosen locations have been constructed using a shortest path tree (SPT) algorithm [5]. This means that for a given number of receivers, 100 trees have been created, which results in a total of 800 trees. In order to give an impression of the structure of the used trees, Figure 4 shows the link delay distribution of a 140 receivers tree used for the simulations.

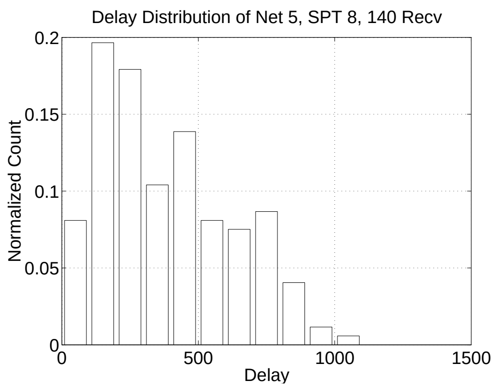


Figure 4: Link delay distribution of a sample tree with 140 receivers

The three center placement algorithms CPSS, CPSM and CPDP were executed on every SPT

- *with* respecting the cluster size constraint (with CSC)
- *without* respecting the cluster size constraint (without CSC)

For the optimal cluster size Z the values 5, 10, 15, 20, 25 and 30 have been used, for the center delay cd the values 100.0, 200.0 and 300.0, resulting in 18 different parameter sets.

All values presented in the following sections are mean values of 100 evaluated trees. Indicated intervals are 95% confidence intervals [10].

4.1 Evaluation Metrics

In order to compare different algorithms, an exact quantification is needed. As the effects of the cluster size and the maximal receiver delay to a multicast connection are not known exactly, they will be treated separately. For the performance evaluation the following metrics are used:

Definition 4.1 Cluster Size Error CSE
The **cluster size error** $\boldsymbol{CSE}$ is defined as

$$CSE = \frac{\sum_{i=1}^{NOC}(Z - cs(c_i))^2}{Z^2 \cdot NOC} \tag{7}$$

Definition 4.2 Delay Increase Cost DIC
The **delay increase cost** $\boldsymbol{DIC}$ is defined as

$$DIC = \frac{mDR_{with\ centers} - mDR_{without\ centers}}{mDR_{without\ centers}} \tag{8}$$

While the cluster size error CSE represents the normalized mean squared error of the cluster size (MSE), which is defined in Definition 2.7 and linked to the accumulated squared error of the cluster size ASE (Definition 2.6), the delay increase cost DIC represents the relative increase of the maximal receiver delay mDR (Definition 2.9).

In the next sections, the proposed algorithms will be evaluated using the above introduced quality metrics.

4.2 Split and Shift (CPSS)

4.2.1 Number of Centers NOC

Figure 5 shows the distribution of the distances in hops between the tree's root and the centers for 40 receivers trees, $Z = 20$ and $cd = 100.0$. It clearly shows that CPSS tends to locate centers further away from the root than do CPSM and CPDP. Note that the distance 0 is caused by the center located on the root. The lower normalized count for distance 0 for CPSS in contrast to CPSM and CPDP results from the higher number of centers for CPSS in general (see also Figure 9).

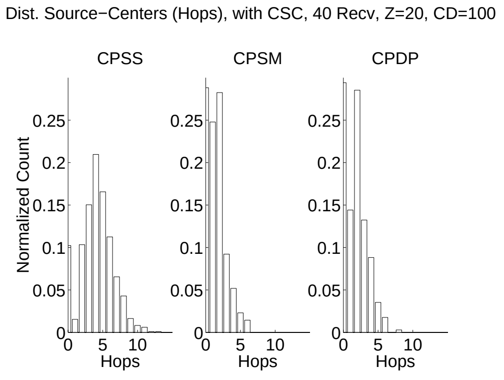


Figure 5: Distribution of distances from source to centers in hops. CPSS, CPSM and CPDP with cluster size constraint

Figure 6 shows for CPSS the number of centers that lie on the paths from the root to the receivers for 40 and 140 receivers trees ($Z = 10$, $cd = 100.0$).

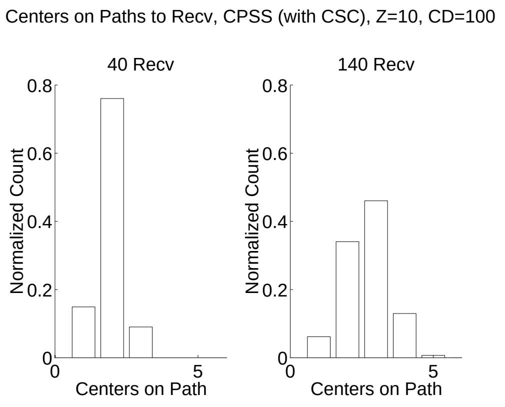


Figure 6: Centers on the paths to receivers, Center Placement by Split and Shift (CPSS), with cluster size constraint, 40 and 140 receivers

4.2.2 Cluster Size Error CSE

CPSS suffers from the fact that numerous clusters near the leaves are sparsely filled. Figure 7 shows the cluster size distributions for the three algorithms CPSS, CPSM and CPDP with cluster size constraint (CSC) for 140 receivers and an optimal cluster size of $Z = 10$ ($cd = 100.0$). Conspicuous

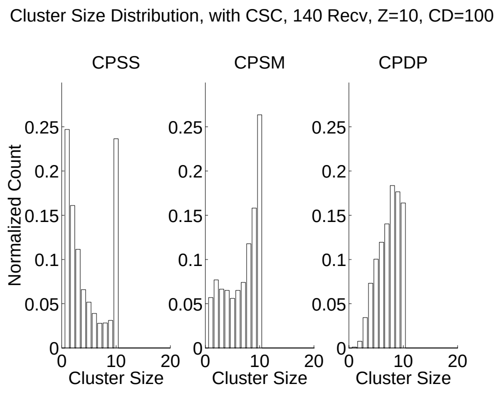


Figure 7: Cluster size distribution for CPSS, CPSM and CPDP with cluster size constraint

are the shapes of the distributions. CPSS often succeeds in reaching the optimal cluster size Z. The high portion of sparsely filled clusters emerges from those clusters located in the leaves areas of the trees. For 140 receivers, $Z = 10$ and $cd = 100.0$, 63.7% of all clusters created by CPSS have a size of $cs(c_i) \leq 5 = \frac{Z}{2}$ (for comparison: CPSM: 32.2%, CPDP: 21.6%).

4.2.3 Delay Increase Cost DIC

In section 2, it has been pointed out that the maximal receiver delay mDR plays a key role in the performance of reliable multicast connections. The relative increase of the mDR (Definition 4.2) is regarded as the basic criterion for evaluating the quality of the placement algorithms in terms of delay.

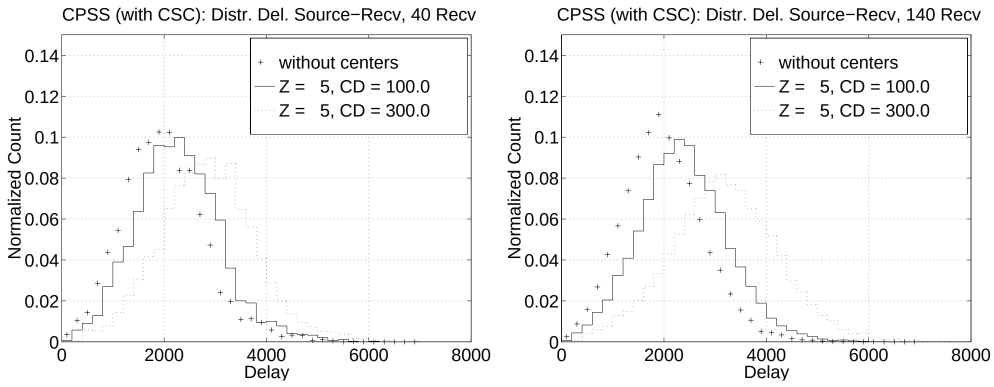


Figure 8: Receiver delay distribution before and after center placement for CPSS with cluster size constraint for 40 and 140 receivers

Figure 8 shows the impact of centers on the general distribution of receiver delays:[5]

- the delays are increased (the shape is moved to the right) and
- the delays are distributed more equally (flatter shape).

4.3 Shift and Merge (CPSM)

Center Placement by Shift and Merge (CPSM) starts placing centers at the leaves of the tree and concentrates primarily on optimizing the cluster size. Consequently, the performance in terms of the cluster size error CSE is very good and outperforms the Center Placement by Split and Shift (CPSS), while the performance in terms of delay is lower than the one of CPSS.

4.3.1 Number of Centers NOC

In contrast to CPSS, CPSM avoids the effect of sparsely sized clusters (see Figure 7). Building clusters from the leaves to the root results in only one really badly sized cluster, the one located at the tree's root. As the average cluster size is closer to the optimal value Z, the total number of clusters in a tree is less than for CPSS (Figure 9).

Note, that the number of centers (NOC) in the trees grows approximately linear with the number of receivers r:

$$NOC \approx k \cdot r \tag{9}$$

with a gradient k inversely-proportional to the optimal cluster size Z:

$$k \sim \frac{1}{Z}, \ \frac{r}{Z} \gg 1 \tag{10}$$

[5]In the figure, receiver delays have been put into bins with a width of 200.0

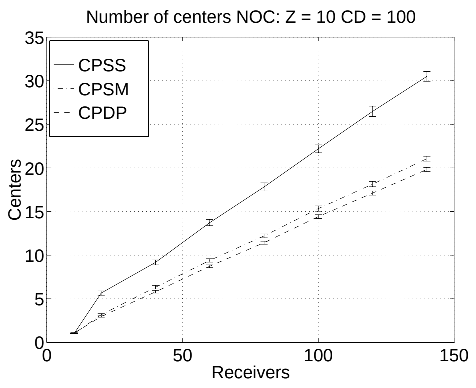


Figure 9: Number of centers NOC. CPSS, CPSM and CPDP with cluster size constraint

4.3.2 Cluster Size Error CSE

CPSM performs much better than CPSS in terms of the cluster size error CSE (Figure 10). CPSM prefers to merge clusters instead of just expanding clusters along the longest path (in terms of delay), like CPSS does.[6]

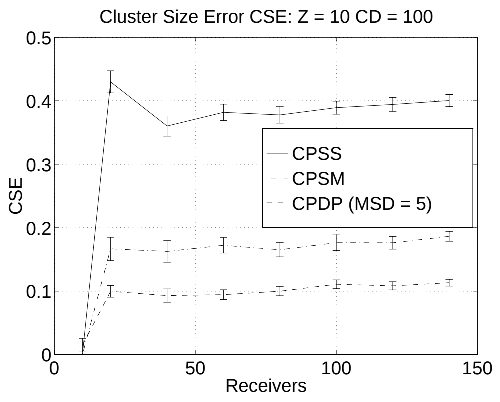


Figure 10: Cluster Size Error CSE. CPSS, CPSM and CPDP with cluster size constraint

4.3.3 Delay Increase Cost DIC

Comparing CPSS and CPSM (Figure 11), the better delay performance of CPSS can be noticed, which is in harmony with the objective of the algorithms: While CPSS tries to optimize the delay criterion (DIC), CPSM optimizes the mean squared error of the cluster size (CSE criterion).

In contrast to the CSE criterion, which offers approximately constant values for different tree sizes, the DIC criterion produces lower values for smaller trees than for bigger ones. This effect cannot be attributed to the algorithms, but is the result of the testing environment: Random networks with $|V| = 200$ nodes have been used for the simulations. The larger the trees get, the higher the ratio receivers/nodes becomes. In total, more centers are placed in the trees, also on the mDR-paths, and the performance is decreasing.

[6]Note that this strategy can also be altered easily (for example, in order to put more emphasis on the delay criterion) by modifying the move-rules described in section 3.2.

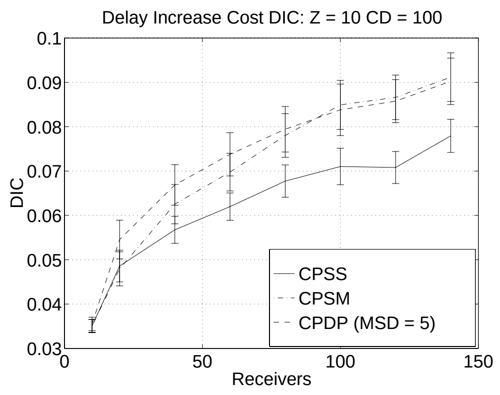


Figure 11: Delay Increase Cost *DIC*. CPSS, CPSM, CPDP with cluster size constraint (CSC)

4.4 Dynamic Programming (CPDP)

CPDP uses the method of dynamic programming. Instead of performing a *full* search on all nodes of the multicast tree for finding the optimal center locations, the amount of time needed to find a good solution is reduced by performing a *local* search, not considering combinations that are unlikely to lead to a good result.

The next paragraphs will show that there exists a close correlation between the extent of the local search and the quality of the resulting placement.

4.4.1 Number of Centers *NOC*

Comparing CPSM and CPDP (Figure 9), a slightly lower number of centers can be observed for CPDP. This little difference needs no further explanation, since the exact number of centers placed by CPDP depends on the placement metric and thus might differ in both directions when the weights of the different criteria are altered.

CPDP's optimization function can be tailored by weighting the two criteria, cluster size and the delay.

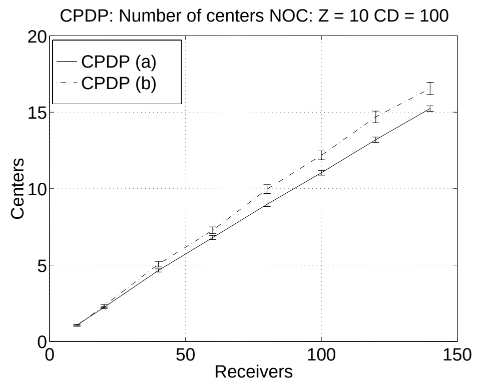
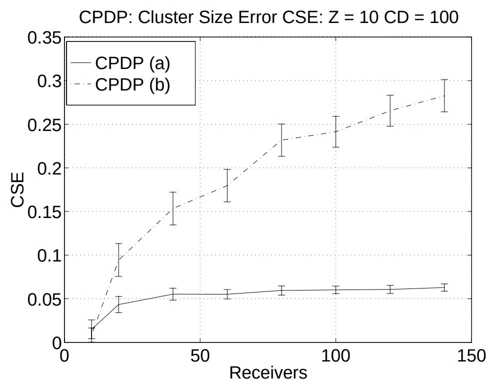
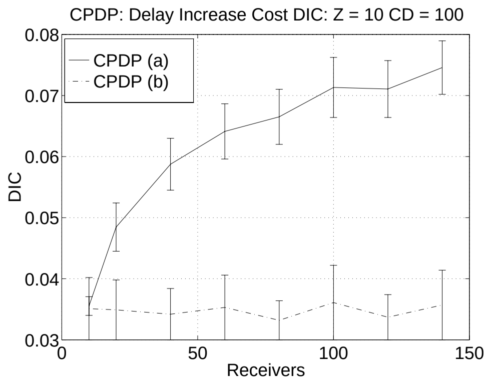


Figure 12: Number of centers *NOC*. Center Placement by Dynamic Programming (CPDP), different weights for the two criteria

While CPDP (a) in Figure 12 weights both criteria equally (0.5 and 0.5), CPDP (b) uses a weight of 0.9 for the delay criterion and only 0.1 for the cluster size criterion. Giving more emphasis to the delay criterion results in a slightly higher number of clusters, a higher cluster size error CSE and a much lower delay increase cost DIC (Figure 13).



Cluster Size Error CSE



Delay Increase Cost DIC

Figure 13: Cluster Size Error CSE and Delay Increase Cost DIC. CPDP (a) and CPDP (b), without cluster size constraint

4.4.2 Cluster Size Error CSE

Figure 14 shows the quality of the placement for CPDP without cluster size constraint (CSC) in terms of the cluster size error CSE for $Z = 5, 10, 20, 30$. We see a high correlation between the optimal cluster size Z and the required maximal search depth without improvement, MSD. While for $Z = 5$ $MSD = 2$ suffices[7] (Figure 14 (a)), there exists already a big penalty in the quality between a MSD of 2 and 4 for $Z = 10$ (Figure 14 (b)). Conspicuous is the fact, that for $Z = 10$ (Figure 14 (b)) the mean squared error of the cluster size in the trees with 10 receivers does not equal 0, which would be indicated by a CSE value of 0. This shows that the best solution, the one with all 10 receivers in one cluster of size 10, is not always found if the maximal search depth MSD is chosen too low.

5 Conclusions

In the last sections, three center placement algorithms have been presented and evaluated. Two of them (CPSS and CPSM) start placing centers on either end of the tree and then proceed consecutively to the other end. The third one (CPDP) uses dynamic programming combined with a local search.

CPSS performs very well in terms of delay, but does not try to optimize the cluster sizes resulting in a high cluster size error CSE. CPSM performs very well in terms of the cluster size error CSE, its delay increase cost DIC is worse than the one of CPSS.

As long as the maximal search depth MSD is chosen high enough, CPDP has a good performance, both in terms of delay and cluster size, as its optimization function can be altered very easily to meet the requirements. However, CPDP has a high computational cost which grows exponentially with the optimal cluster size Z.

[7] There is about no difference between the two lines for $MSD = 2$ and $MSD = 4$.

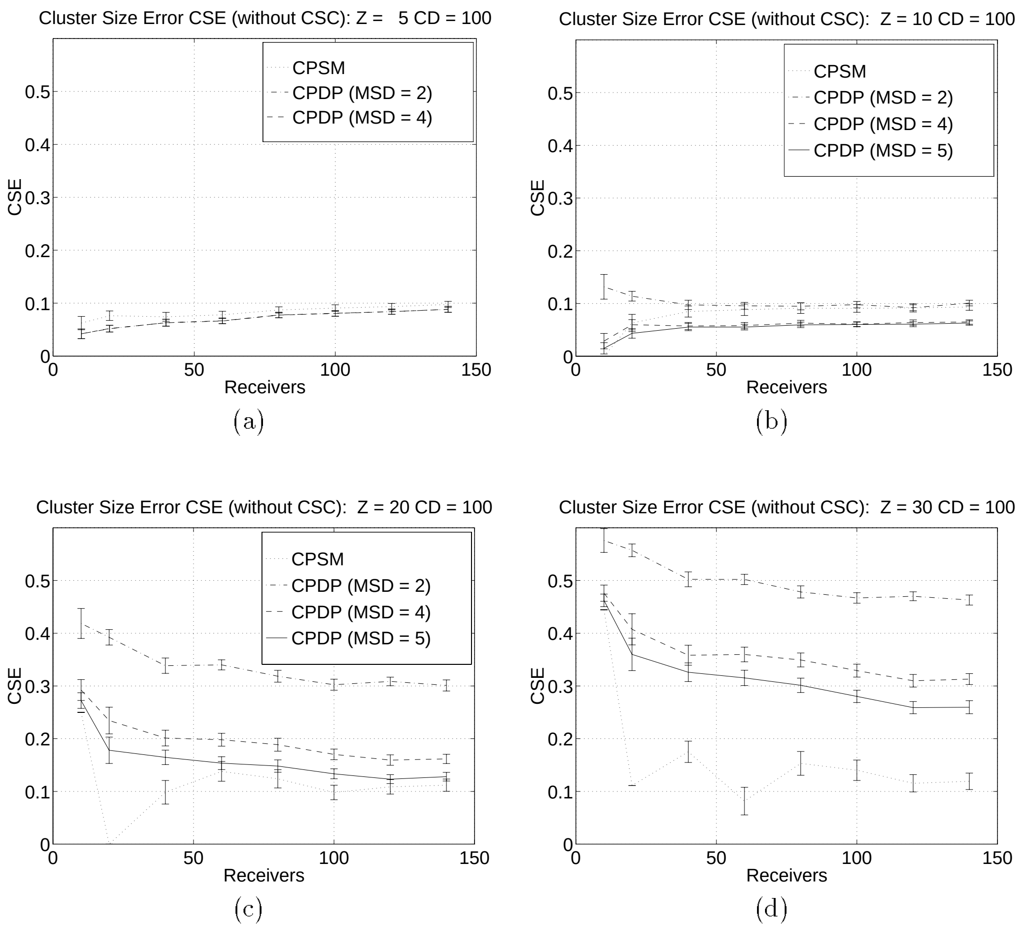


Figure 14: Cluster Size Error CSE for CPSM and CPDP without cluster size constraint

The results obtained lead to the following conclusions:

- For small optimal cluster sizes ($Z \in [5 \ldots 10]$), CPDP is recommended, since its optimization function can be altered easily to meet predefined requirements.
- For bigger values of Z, CPDP's complexity is too high and preference should be given to CPSS or CPSM.
- If delay aspects have highest priority and cluster sizing only plays a minor role, CPSS should be used.
- If cluster size is important, CPSM should be used as it obtains small values for CSE and has a low computational cost.

A Waxman Graphs

Waxman graphs [17] are constructed by randomly placing n nodes in a Cartesian grid. Then, for each pair of nodes u, v an edge is placed with the probability:

$$P_k(u,v) = \beta \exp\left(\frac{-d(u,v)}{\alpha L}\right) \tag{11}$$

$d(u,v)$ denotes the Euclidean distance between node u and node v, L is the maximal possible distance between two nodes and α and β are two parameters in the range of $0 < \alpha, \beta \leq 1$. A bigger value of α increases the number of edges of a large distance, while a bigger value for β increases the outdegree of the nodes. For the simulations, the parameters α and β have been fixed to $\alpha = 0.25$ and $\beta = 0.09$. The Euclidean distance $d(u,v)$ between two nodes u and v that are connected by an edge is interpreted as link delay. Figure 15 shows a sample network.

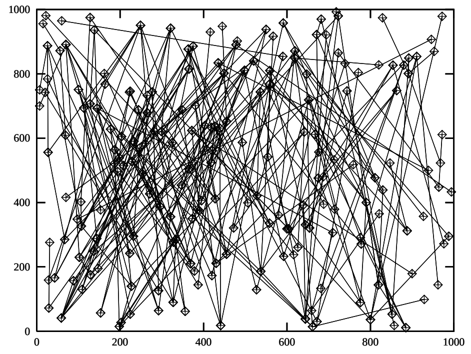


Figure 15: Random network with 200 nodes and an average outdegree of 3.0

References

[1] Azer Bestavros. Speculative data dissemination and service to reduce server load, network traffic and service time for distributed information systems. In *Proceedings of ICDE'96: The 1996 International Conference on Data Engineering*, New Orleans, Louisiana, USA, March 1996.

[2] D. R. Cheriton. Dissemination-oriented communication systems. DSG Working Paper, 1992.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[4] S. Deering. Host extensions for ip multicasting. *Internet Request for Comments*, RFC 1112, August 1989.

[5] M. Doar and I. Leslie. How bad is naïve multicast routing. In *Proceedings of INFOCOM'93*, volume 1, pages 82–89. IEEE, 1993.

[6] Hans Eriksson. MBONE: The multicast backbone. *Communications of the ACM*, 37(8):54–60, August 1994.

[7] Sally Floyd, Van Jacobsen, Steven McCanne, Ching-Gung Liu, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of SIGCOMM'95*, volume 25, pages 342–356. ACM, October 1995.

[8] Matthias Grossglauser. Optimal deterministic timeouts for reliable scalable multicast. In *Proceedings of INFOCOM'96*, volume 3, pages 1425–1432, San Francisco, CA, USA, March 1996. IEEE.

[9] Markus Hofmann. A generic concept for large scale multicast. In *International Zürich Seminar*, Zürich, Switzerland, February 1996.

[10] Robert V. Hogg and Allen T. Craig. *Introduction to Mathematical Statistics.* Macmillan Publishing Co., Inc., New York, 1989.

[11] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of SIGCOMM'95*, volume 25. ACM, October 1995.

[12] Vachaspathi P. Kompella, Joseph C. Pasquale, and George C. Polyzos. Multicasting for multimedia applications. In *Proceedings of INFOCOM'92*, pages 2078–2085. IEEE, 1992.

[13] John C. Lin and Sanjoy Paul. Rmtp: A reliable multicast transport protocol. In *Proceedings of INFOCOM'96*, volume 3, pages 1414–1424, San Francisco, CA, USA, March 1996. IEEE.

[14] Ch. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[15] Sridar Pingali, Don Towsley, and James F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *Proceedings of Sigmetrics, Conference on measurement and modeling of Computer Systems*, pages 221–230, Santa Clara, CA, USA, May 1994. ACM.

[16] Macedonia. M. R. and D. P. Brutzmann. Mbone provides audio and video across the internet. *IEEE Computer*, April 1994.

[17] B. M. Waxman. Routing of multipoint connections. *IEEE JSAC*, 6(9):1617–1622, December 1988.

[18] Liming Wei and Deborah Estrin. The trade-offs of multicast trees and algorithms. In *Proceedings of ICCCN'94*, San Francisco, CA, USA, Sept 1994.

[19] T. Yan and H. Garcia-Molina. SIFT - A tool for wide-area information dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, pages 177–186, 1995.

[20] Rajendra Yavatkar, James Griffoen, and Madhu Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of ACM Multimedia*, pages 333–344, San Francisco, CA USA, 1995. ACM.