Peer-to-Peer Networking and Applications (2019) 12:724–739
https://doi.org/10.1007/s12083-018-0668-7

Hierarchical P2P architecture for efficient content distribution

Rogelio Hasimoto-Beltran[1] · Francisco de Asís Lopez-Fuentes[2] · Misael Vera-Lopez[2]

Received: 31 August 2017 / Accepted: 6 July 2018 / Published online: 1 August 2018

Abstract
Mutualcast is a one-to-many (peer-to-peer) scheme for content distribution that maximizes the overall throughput during a broadcast session. It is based on a fully-connected graph (full mesh topology), which introduces benefits such as robustness or simultaneous transmission from/to multiple devices. The main disadvantage of Mutualcast is scalability; it is constraint to a small P2P group for content distribution. In this paper, we make Mutualcast scalable. We propose a highly collaborative and scalable P2P tree-based architecture made of two main components: 1) Peer grouping or clustering and 2) Hierarchical tree-based content distribution. In step 1), peer nodes (content receivers) are grouped into equal-size clusters by using a proposed heuristic size-constrained algorithm based on k-means. In step 2), clusters (which become the nodes of the tree) are organized into a single hierarchical n-ary tree-based architecture, in which the root of the tree (Root Cluster) is the one closest to source peer, while intermediate and leaf clusters are positioned in the tree according to their delay-proximity to previously inserted clusters. During content distribution, the root cluster receives the blocks of content before any other cluster in the tree and directly from (and only from) the source peer; blocks are then passed on to the next hierarchical level down the tree in order (higher levels of the tree receive the content before lower levels). Inter-clusters and intra-clusters content distribution is performed concurrently and takes into account peers upload/download capacities to relay blocks of content. The evaluation of our hierarchical P2P architecture concentrates on the following metrics: scalability of the systems, overall end-to-end delay distribution, and efficient cluster size. Finally, our architecture is compared against two well-known P2P technologies in the literature, Super-Peer and Kademlia.

Keywords Peer-to-peer networks · Video multicast · Content distribution · Clustering

1 Introduction

Multimedia content distribution over the Internet has increased at a very fast rate with significant impact on today's global economy. Popular services such as videoconferencing and Internet Protocol Television (IPTV) markets are expected to reach USD 7.94 and 95.9 global Billion by the 2020s, respectively [1, 2]. These services make use of media multicast technologies where information is addressed to a group of destination computers simultaneously using one-to-one (unicast) or one-to-many (multicast) schemes. Research teams in academia and industry worldwide are making significant efforts to innovate multicast architectures to address the challenges of a rapidly increasing market.

✉ Francisco de Asís Lopez-Fuentes
flopez@correo.cua.uam.mx

[1] Centro de Investigación en Matemáticas, A.C, Guanajuato, Mexico

[2] Universidad Autónoma Metropolitana, Mexico City, Mexico

Traditionally, Internet Protocol Multicast (IPM) has been proposed as an efficient solution for one-to-many media dissemination [3]. IPM is more efficient than unicast due to its reduced transmission overhead from the sender to all receivers. IPM decreases traffic by simultaneously distributing a single copy of data packets to thousands of users through networks routers. However, IPM has not been fully deployed in the Internet due to network control and management issues raised by Internet Service Providers (ISP). Thus, the deployment of IP multicast is currently limited to local area networks, and a handful of ISPs networks [4]. To address these issues, researchers have proposed an application level solution as an alternative to implement IPM [5, 6]. For example, in Application Layer Multicast (ALM), all tasks are implemented by collaborative work in the end-users exclusively, while the network infrastructure is kept fixed. ALM approaches provide more flexibility and are easier to deploy than those requiring network router multicast support.

In addition to ALM technology, peer-to-peer (P2P) computing technology has emerged as a novel paradigm to face some of the limitations of the client-server model [7]. The end

users provide all the communication infrastructure needed, so dedicated infrastructure is not required. Each user provides a communication node, and all the nodes comprise a network abstraction on top of the physical network known as an overlay network, which is independent of the underlying hardware network implementation. In the P2P systems, each peer can take the role of both server and client at the same time, so there is no need for dedicated servers. Due to the sharing of peer resources, the ALM scheme is an effective means for conducting the cooperative P2P communications. Namely, during a multicast session, peers contribute their resources to relay the media to others. In this way, as a new peer arrives to the P2P system, the demand is increased, but the overall capacity also increases. This feature is not available in a system based on a client-server model. In a P2P multicast system, the media must be delivered to all requesting peers with high quality and minimal delay. An overlay P2P multicast does not require any router support and is the most flexibility and adaptable to diverse requirements from these applications.

Most P2P multicast implementation algorithms can be classified according to the data structure used to support packet distribution (i.e. trees, forests, or fully-connected graph) [5]. In conventional tree-based distribution algorithms, the peers placed as interior nodes redistribute data content, while the peers placed as leaf nodes only receive data. Although the multicast-tree based scheme is highly scalable [8–10], it is not maximally efficient in collaborative environments, because the upload capacity of the leaf peers is not used during a multicast session. The full upload capacity of all participating peers is required in order to achieve maximum throughput. A possible solution to increase efficiency consists on constructing multiple concurrent trees, where peers contribute with their upload capacity in at least one tree or in the construction of a fully-connected network. A drawback of using a fully-connected graph (or mesh architecture) is that the number of connections is proportional to the number of peers, because each peer has to forward its received blocks from source to all other peers. Mesh-based approaches also have high control overhead due to data scheduling and limitations for delay sensitive applications when the participating peers are located in different geographical locations. On the other hand, the dynamic behavior of peers in P2P systems is one of the major challenges. Since peers are transient in nature, once a parent peer departs from the multicast system [5], the receivers receiving streaming content from that parent peer might suffer a temporal interruption in the content transmission.

In this paper, we propose a fully collaborative and scalable P2P architecture which involves strong cooperation between participating peers during the content distribution from a source to multiple peers. Participating peers are organized into different clusters or groups based on delay-proximity. Peer delay-proximity is exploited in our proposed scheme in order to form a fully hierarchical cluster of peers interconnected via a single n-ary tree [11], with excellent content propagation time. The source-peer (root of the tree) divides the content into blocks and distributes different blocks to all peers in the highest hierarchical cluster (root cluster), so that each peer within the cluster contributes its redistribution capacity by forwarding the receiving blocks to the rest of peers within its own group and receiving at the same time the rest of blocks not directly obtained from the source peer. An n-subset of peers within the cluster is designated as source for the n lower clusters in the next set of clusters down the hierarchy tree structure, that is, one peer is designated as source for each receiving cluster. The process continues in the same way, until all cluster leaves are reached. We evaluate our proposed architecture based on the overall end-to-end delay distribution to all peers, tree-based scalability, and cluster size. A comparison against two well-known P2P technologies in the literature, such as Super-Peer [12] and Kademlia [13] is presented.

The remainder of this paper is organized as follows. We introduce and discuss some collaborative multicast approaches in Section 2. We briefly explain how to build the collection of clusters connected via a simple tree in our proposed architecture in Section 3. How the collaborative architecture is implemented in the simulator is explained in Section 4. In Section 5, we evaluate the performance of our collaborative architecture against other content distribution schemes. Section 6 concludes the paper.

2 Collaborative multicast schemes

In this section, we describe the main technologies our scheme is based on: mesh-based approaches (such as Mutualcast) and tree-based approaches. Mutualcast has shown to be a scheme that maximizes the overall throughput during a multicast session. In addition, Mutualcast is based on a fully-connected graph (full mesh topology), which introduce benefits such as robustness or simultaneous transmission from multiple devices. On the other hand, a tree-based scheme introduces several benefits such as scalability, reduced end-to end delay and easy maintenance. Our aim is to reach shorter end-to-end delivery time, improve scalability and low resources consumption by merging these two technologies into an efficient content distribution scheme.

2.1 Tree-based approaches

In a tree-based approach, an overlay construction mechanism organizes participating peers into a single tree whose root is located at the source node. The participating peers are organized into a single tree following their classification as interior node or leaf node. In a tree-based topology, the source peer sends the data to the requesting peers on the first level, which then forward the data to the requesting peers located on the

following level down the tree structure and so on until reaching the leaf peers. In this configuration, a video stream is pushed from a parent router to its children routers along a well-defined route. In this way, the multicast tree for content distribution uses the upload capacity of the peers located on the intermediate levels. However, the upload capacity of the leaf peers is not used. Although a tree approach probably represents the most effective distribution structure in terms of bandwidth and delay optimization [14], this configuration has an inherent drawback because all the burden generated by forwarding multicast messages is carried out by a relative small number of interior nodes.

2.2 Mesh-based approaches

In a mesh-based overlay, a peer can concurrently receive data from different senders, each contributing a portion of its upload capacity. Additionally, the requesting peers can send and also receive data from each other. Video data in a mesh-based P2P multicast is available in multiple neighboring peers, with a node having to pull data to avoid significant redundancies, while in a forest based overlay the data is pushed from a parent peer to many child peers. Due to the dynamic and unpredictable behavior of peers, the main challenge of a mesh-based overlay is how to select the proper senders [15] and how to cooperate and schedule the received data in the requesting peers. In a collaborative environment such as a P2P network, the participating peers contribute with resources proportional to the benefits they obtain from the system. Specifically, in an application layer multicast, the peers expect that the forwarding load will be shared among all participants [6]. However, a multicast based on a single tree does not match well with these cooperation expectations, because a small number of interior peers carries the forwarding multicast traffic, while the upload capacities of a large number of leaf peers are not used. This is a critical problem for applications with high bandwidth requirements such as video or bulk file distribution, because many interior nodes in the multicast tree may not have the required upload capacity. To face these challenges, our proposed scheme adopts a tree structure as the global structure but incorporates small mesh clusters on each level of our single distribution tree. Clusters are an elementary unit in this hierarchical architecture, which involves one source peer and several requesting peers. The peers inside a cluster are fully connected, and each peer inside a cluster is a receiving and forwarding peer at the same time. Due to the fact that the upload capacity of all peers is also used, the bandwidth consumption from the source can be reduced.

2.3 Hierarchical clustering approaches

Tree and mesh overlay topologies have been found not suitable for large scale dynamic P2P networks; they become inefficient and involve high control overhead. The concept of hierarchical clustering has emerged as a new alternative in which, peers are grouped into clusters and clusters into an organized tree topology. In NICE scheme [9], a balanced tree of clusters is built, in which all peers are part of the lowest layer including the source peer. Higher layers of the tree are represented by corresponding cluster centroids of lower layers, in this way the root of the tree is the centroid of all cluster centroids of lower layers. This model simplifies the insertion of peers in the hierarchical tree. NICE uses the head to forward the content to its subordinates, thus incurring a high bottleneck. Additionally, NICE tree-structure is fixed and not optimal; it does not provide the best low-latency distribution tree. Broadly speaking, it becomes a special case of our proposed hierarchical scheme. An extension of NICE is presented in [16], called Zigzag protocol. It is derived from the same balanced multicast tree developed in NICE, with a modified intra-cluster communication strategy. In this new strategy, intra-cluster peer communication is not allowed and each peer must relay completely to subordinate cluster or peers. Zigzag extends the nomenclature of the administrative organization of peers, claiming a reduced control overhead compared to NICE. One of the main drawbacks of Zigzag is peer-insertions, which occurs whenever there is place available in a cluster, affecting the transmission delay. A more recent scheme named TURINstream [17], combines a tree structured P2P video streaming scheme with Multiple Description Coding (MDC) to achieve low-delays, robustness to peer dynamics and limited protocol overhead. In MDC, video is composed by independent and complementary descriptions which can be decoded independently, yielding the base video quality (the more descriptions are received the better the quality of the video). The advantage of MDC is playback continuity despite peers´ departures, failures, and churning. The algorithm for building the tree is very simple, clusters must provide the upload capacity for a continuous transmission; it does not pay attention to optimal transmission delays efficiency. This is the main problem of TURINstream; a peer can be joined at any level of the network, it just needs to follow a path along the control tree until it finds a cluster that can host it (just based on the upload capacity).

Our scheme, is focused on improving the deficiencies of the above algorithms by building a new hierarchical tree topology that improves transmission efficiency in several ways: it reduces upload bandwidth usage, peer communication stress, and increases transmission robustness.

3 Proposed approach

Our underlying ground on proposing a new scalable scheme is that peers can greatly benefit from the capacity of other requesting peers via collaboration. Collaboration becomes a key factor for efficient multicast applications over large-scale heterogeneous environments. Based on these facts, we focus on developing a

collaborative computing system considering both the dynamic behavior and scalability of the networks. To achieve this goal, our proposed architecture is mainly constructed with a superposition of two overlapping networks, one using the tree model, which is the main structure (the body of the architecture), and the other using the mesh model (Mutualcast [18]).

3.1 Tree distribution model

Mesh-based P2P multicast (such as Mutualcast) can achieve the maximum overall throughput but incurs scalability limitations because all nodes are fully connected. To deal with these limitations, our proposed architecture uses clusters of peers allocated in a unique tree-rooted distribution at the source node. The hierarchical structure of our approach is shown in Fig. 1. The first level of the network hierarchy is the peer source (root node) that contains the original file. Initially, active peers in the system are grouped into small clusters (see section 3.2 for details), ensuring that peers closest to the root node (source peer S) will form the root cluster in the distribution tree (cluster 1). Peers with longer time proximity to the source peer are grouped as intermediate and leave clusters in the hierarchical tree. Leaves (cluster 2–4) have the longest time proximity to source peer. In this work, we consider clusters with the same number n of peers, but it can be easily extended to unequal clusters size (as a future work). In Fig. 1, peers P1, P2 and P3 form the cluster with the highest hierarchy level in the multicast session, while the rest of clusters and corresponding peers are subordinates. That is, information is first distributed from the source node to the root cluster, and from the root cluster to the rest of clusters following a sequential top-down order along the tree. Each peer forwards the blocks received from the source to the rest of the peers within the same cluster, and simultaneously receives the rest of the blocks from the other peers in the cluster. Peers in the same cluster share bidirectional communication. Concurrently, each peer in the first cluster acts as a source for a new cluster located on the second level of our hierarchical structure. Thus, peer P1 is a forwarding peer of cluster 1 and a source peer of cluster 2 (which is formed by peers P4, P5 and P6) at the same time. Peers P2 and P3 can also extend their own clusters. We denote cluster 2 as a child-cluster of peer P1.

The communication between peers located in the first cluster and the requesting peers clustering on a second level is unidirectional. In other words, in the hierarchical approach, the blocks are distributed from one cluster to another, in a top-down fashion. Using clustering, the peers can greatly benefit from the capacity of other neighboring requesting peers via local collaboration while the number of connections is reduced in comparison to a fully connected overlay topology. The total number of connections TC (for a constant cluster size) in our hierarchical scheme can be represented by:

$$TC = k*\left[\frac{(n)(n-1)}{2}\right] + l*n + \left[\frac{(p)(p-1)}{2}\right] \quad (1)$$

[internal external residual]connections

where $n = \lfloor N/k \rfloor$ represents the cluster size, N is the number of peers in the system, k is the number of clusters, l is the number of links in the tree (external node-to-node + source-to-root connections), and remaining peers $p = (N \bmod k)$ are allocated *in* a final *p-size* cluster. In a multicast group with $N = 150$ requesting peers, $k = 30$, $n = 5$, and $l = N$, our proposed architecture needs 1050 connections to distribute all the blocks. In contrast, using a fully-connected architecture (e.g. Mutualcast [18]), the overlay network is formed with $(N-1)(N-2)/2 = 11026$ connections. In this way, our proposed architecture is scalable and robust at the same time. The overall delay optimization problem for minimizing the content distribution time is more complex than just considering the number of connections. It involves N, n, peers´ upload and download capacity, and the final structure of the distribution tree. In the next and

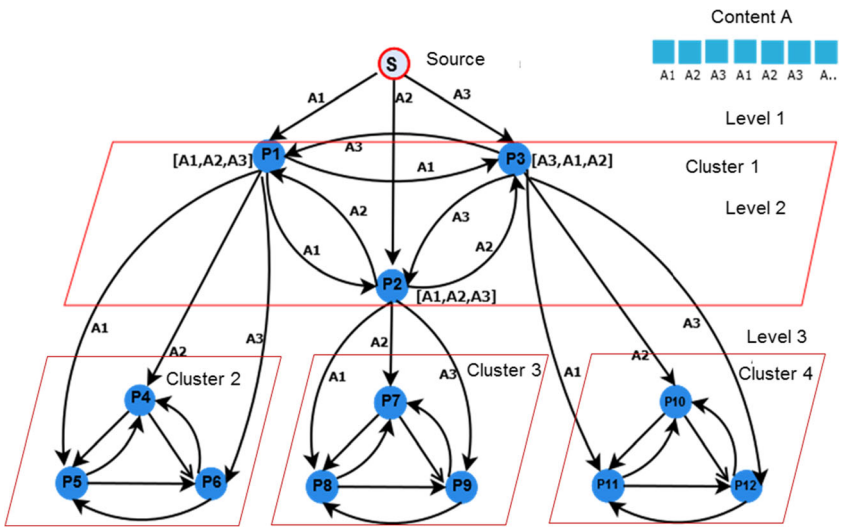

Fig. 1 Scalable collaborative multicast

experimental sections, we will address this problem in detail. For the moment, we concentrate on a very important step in our scheme, peer clustering.

3.2 Constraint clustering process

In this work, we use the Round-Trip Time (RTT) between two peers as proximity information to build the local clusters. Given a data set $RTT_i = \{rtt_{i,\ 1}, rtt_{i,\ 2}, \ldots, rtt_{i,\ n}\}$, $i = 1, \cdots, N$, representing the Round-Trip-Time (RTT) from the i^{th} peer to all active peers in the system, our aim is to partition the *N(N-1)* observations into k mutually exclusive clusters $S = \{S_1, S_2, \cdots, S_k\}$ that minimize the sum of squares (within the cluster) given by:

$$arg\ \min_{S} \sum_{j=1}^{k} \sum_{rtt_{l,m} \in S_j} \left(rtt_{l,m} - \mu_j \right)^2 \tag{2}$$

where μ_j is the centroid of the cluster S_j, whose cardinality is $|S_j|$. The solution of Eq.2 for a global minimum is an NP-Hard problem, since there exist $k^N/k!$ different ways for grouping an *N(N-1)* data set [19]. Instead, several heuristics have been developed to provide local minimums or suboptimal solution to this problem, the simplest and most widely known is the k-means or Lloyd's algorithm. Lloyd's algorithm is based on the simple observation that the optimal placement of a center is at the centroid of the associated cluster. The algorithm proceeds as follows [20]:

Algorithm 1: k-means

1. Select k random centroids for the initial partition of the data space.
2. Assign each data point *rtt* to the cluster corresponding to the closest centroid:
   a. For each cluster centroid μ_j, compute the distance between $rtt_{\mu_j,i},\ j = 1, \cdots, k;\ i = 1, \cdots, N^2$.
3. Calculate the new centroid of each cluster.
4. Repeat steps 2 and 3 until the algorithm converges (centroids do not change anymore).

The k-means baseline algorithm has been modified to satisfy the following constraints imposed in our hierarchical tree-base model: **a)** the centroid of a cluster must always be a peer; **b)** RTT values are nonsymmetrical; and **c)** the number of peers in the clusters must be small and constant. The first constraint avoids the use of fictitious peer centroids for which we cannot measure RTT distances from/to any peer (because of the non-linearity in the data). A major benefit of this constraint is faster convergence time since the algorithm does not need to re-compute the RTT values for each iteration as in the original k-means. The second constraint (complements the first constraint) takes into consideration that RTT is not symmetric, that is the distance from A to B is not necessarily the same as the distance from B to A. Since RTT values are 1-D scalars without intrinsic spatial distribution information, the way to recalculate the new centroid position of cluster S_l in step 3 of algorithm 1, is by finding the peer for which:

$$\min\left(\sum_{j=1}^{S_l} rtt_{i,j} \right), \forall i, j \in S_l \tag{3}$$

as depicted in Fig. 2. Let us consider that all peers in S_l can be spatially located as shown in Fig. 2a (this is not possible in the real world), and the distance from peer *i* to all peer *j's* can be schematically represented as shown in Fig. 2b. It is easy to see that the minimum of Eq. 3 corresponds to peer $i = 5$, which becomes the new centroid of the cluster. Figure 3 shows the final clustering (one instance of many possible) of the k-means for $N = 20$ peers, $k = 4$ clusters and $n = 5$. Figure 3a shows the initial peer distribution and Fig. 3b the corresponding clustering output.

The last constraint mentioned above (constraint c), is related to Mutualcast mesh connection limits, which is approximately <15 nodes. There exist good solutions to this problem in the literature (see [21]) with increased time and implementation complexity (requires the use of linear programming). Instead, we developed a simple heuristic approach to satisfy the cluster size constraint once k-means is applied (our scheme takes advantage of the sub-optimal k-means output). It is worth mentioning that more than finding an optimal partition of peers, our main contribution is the hierarchical approach, in which information is being transmitted in both ways, horizontally and hierarchically vertical at the same time. Given the output of algorithm-1 and cluster size $|S_j| = n = N/k, j = 1, \ldots k$, our size-*constrained algorithm* consists of the following steps (If $n = N/k$ is not integer, one of the cluster will have $n + (N \bmod k)$ peers):

Algorithm 2: Cluster size constraint

Input Data: k-means clustering (Algorithm-1)
Output Data: n-constrained peer clustering

1. Create a Cluster Distance Array CDA in ascending order using SP as a point of reference (closest clusters to SP are on top of the array).
2. For each cluster Sj , j = 1, . . . , k in CDA:
   a) If |Sj | = n, Sj is done and not considered for further peer exchange process.
   b) If |Sj | >n and j = 1, hand over the n − |Sj | farthest peers Pm (with respect to SP) to the closest cluster Si . Sj is done and not considered member of CDA for further peer exchange process. Update the centroid of S_j and S_i.
   c) If $|S_j| > n$ *and* $j > 1$, pick the closest peers to the centroid (it may include peers from other surrounding clusters) such that $d(P_s, \mu_j)$ is minimum, and hand over the $n - |S_j|$ peers P_h to the closest clusters, such that $d(P_h, \mu_i), m = 1, n - |S_j|; \forall \mu_i \in CDA$ is also a minimum. *d(X,Y)* is the RTT distance from *X* to *Y* and P_s are the peers staying in S_j. S_j is done and not considered member of CDA for further peer exchange process. Update the centroids of all modified clusters.
   d) If $|S_j| < n$, take $|S_j| - n$ peers from the closest cluster S_i, such that the distance $d(P_m, \mu_i), m = 1, j; \forall \mu_i \in CDA$ is a minimum. S_j is done and not considered member of CDA for further peer exchange process.

Generally speaking, algorithm-2 groups peers with minimum distance to the centroid. The exception is the closest cluster to source peer SP, for which it takes those peers closest to SP without considering their distance to the centroid. Figure 4, shows the output of algorithm-2 for $N = 20$, $n = 5$, and $k = 4$.

Clustering process is a centralized process (run by source node or a dedicate server). However, we have also considered its implementation in a distributed system where source node (or a dedicated server) and cluster centroids participate in the process. Arriving peers receive from source node (or dedicated server) an ordered list of all cluster centroids they may join in; peers select its closest cluster centroid and send a join-in request. If the cluster is full, it hands over its farthest peer (including the new peer in the computation) to another cluster, and receiving cluster repeats the peer insertion algorithm. When cluster centroid changes after a peer insertion, parent and children clusters are informed of the new changes (this is important for the forwarding content and control parameters). This

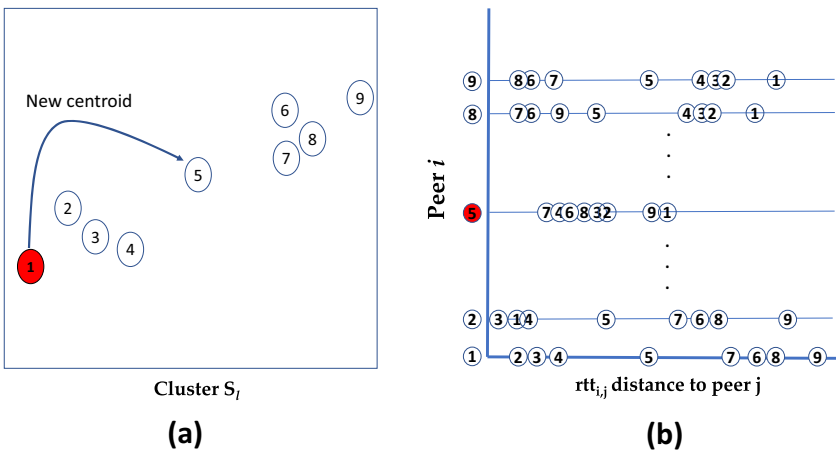

Fig. 2 Locating the new centroid from RTT measures. (a) Original Cluster, and (b) Representation of RTT distance from each peer i to all peers

distributed algorithm is highly scalable (the number of comparisons is at the most half of the number of centroids) and dynamically adapts (for optimal delay) on every single peer insertion.

3.3 N-ary tree creation

In our *n-ary* tree creation process, we favor (if possible) shorter communication links between peers and clusters for robustness, in particular for TCP connections. The tree is created in a top-down fashion with two pre-inserted nodes: the root of the tree or Root Cluster (RC), which is the one closest to the source peer (SP), and the first left child of RC, representing the closest cluster to RC. Hereafter, tree node insertions depend on the relative distance to their closest node and corresponding parent. Since higher hierarchical nodes send data to children, the RTT values considered for the insertion to the tree are from current nodes distance in the tree, to prospect nodes (already computed for the clustering process described in previous section). Let S_j be the next node to be inserted in the tree coming from a predefined node list (LC) ordered from closest to farthest distance with respect to RC, S_i be its closest node (already in the tree), and P the parent of S_i. If $d(P, S_j) < [d(P, S_i) + d(S_i, S_j)]/K$, then S_j becomes a child of P (or sibling of S_i) (if the number of children of P is less than n), otherwise is inserted as child of S_i. A special case during the insertion of cluster S_j is when P is RC; RC could take more than one child if RC has enough upload capacity. This is useful for reducing transmission delay when two clusters are close to RC but in opposite sides. Depending on the value of the constant K, it favors the creation of more balanced trees ($K > 1$), deeper trees ($K < 1$), or no influence at all in the final tree organization ($K = 1$). The example in Fig. 5 shows the following information: centroid distances of the clusters, clusters´ ordered list LC, and S_2, the next cluster to be processed (Fig. 5a). Since $d(RC, S_2) = 5 < 7 = [(d(P, S_i) = 3) + (d(S_i, S_j) = 4)]$ for $K = 1$, S_2 is linked to RC (parent of S_1) creating at this point a 2-level tree (root and two children). For $K = 1.5$, S_2 becomes a child of S_1, creating a 3-level tree, one node per level. The last iteration produces the final 5-ary tree shown in Fig. 5b and c for $K = 1$ and $K = 1.5$ respectively.

The tree creation algorithm is described as follows:

Algorithm 3: *n-ary* tree creation

Input Data: n-constraint k-means clustering (Algorithm-2)

Output Data: Content distribution tree

1. Get the closest distance cluster from SP, it becomes the Root Cluster (RC).
2. Get an ordered distance list LC (closest to farthest) between RC and the rest of clusters. The first cluster in the list becomes RC left child. Clusters will be inserted in the tree following the order of LC.
3. Compute the distance from all nodes in the tree to S_j, the next cluster in the ordered list LC. Let S_i (in the tree) be the closest node to S_j.
   a. If S_i is a child (has a parent P), compute $d(P, S_j)$, $d(P, S_i)$, $d(S_i, S_j)$ and make the following decision:
      i. if $d(P, S_j) < [d(P, S_i) + d(S_i, S_j)]/K$, and $nc < n$, S_j becomes a child of current parent (or sibling of S_i); otherwise S_i becomes a child of S_i.
4. Repeat step 3 until all nodes in LC have been inserted in the tree.

As part of the control topology, every peer manages a peer list with the following information: peer follower, peer source, and peer consumer. Followers are members of the same cluster who alertly watch a predefined partner; in the case of fail, the follower will take over its duties. Figure 6 shows an example of the information or control list carried out by all peers. Peers in the current cluster column watch for themselves, in this case, 1 is the follower of 2, 2 of 3, 3 of 4, and 4 of 1; if peer 3 fails or leaves, peer 2 takes over its functions including receiving from peer 6 (source) and forwarding to peer 5 (consumer). Peer 2 now watch for peer 4 in current cluster. Peers periodically sends keep-alive packets (acks every n packets) to its clustermates. When a peer fails or leaves, the cluster centroid initiates a peer request (in order to maintain the same number of peers) to children clusters. After the handover, the child cluster repeats the same action with its children until a leaf is reached. If current cluster is a leaf, it will stay as is. All peers maintain the last packet correctly received, so when

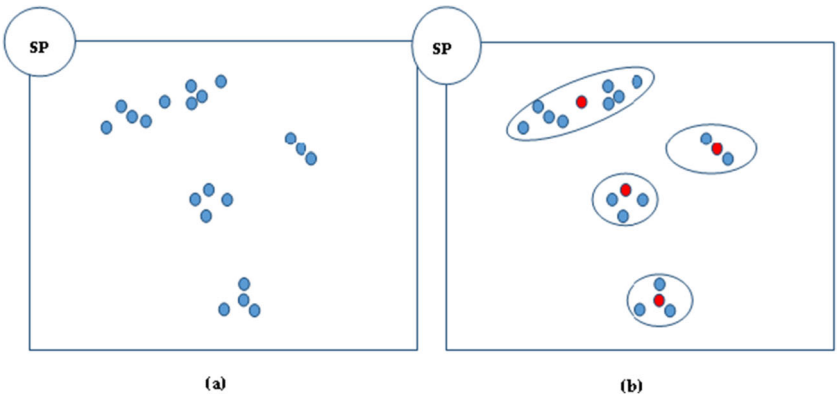

Fig. 3 (a) Original peer distribution, (b) k-means output (centroids are marked in red and SP is the Source Peer)

resuming transmission because of a peer fail, the algorithm ensures that all content will be received.

3.4 Intra-cluster and inter-cluster communication

Once the tree of clusters is created, the source node divides the content (e.g. a file or media stream) into small blocks, to be sent out to the highest hierarchical cluster or root cluster in the tree. Within the root, every peer is designated as receiver from the source node and as sender to the next and closest hierarchical level of clusters (as shown in Fig. 1). Every peer receives different blocks, which are concurrently redistributed to all peers within the cluster (*intra-cluster communication*) and at the same time to the next level of clusters down the tree hierarchy (*inter-cluster communication*). The same process is performed by lower cluster levels until source content reaches all peers in the leave clusters.

Similar to Mutualcast [18], an optimal bandwidth allocation strategy is implemented using redistribution queues between the source and requesting peers. In each cluster, a fully connected topology is built considering proximity information. Within each cluster, content is distributed among all participating peers, which are also called *requesting peers*. Peers are in fact receivers (Re) and senders (Se) at the same time. Each source splits the original content into small blocks and one unique peer is selected to distribute a block to the rest of the peers. Each requesting peer forwards the blocks directly received from a source to the rest of the participating peers in its own cluster. Peers with different upload capacity distribute a different amount of content. When the source peers have abundant upload resources, each source additionally sends one block directly to the receiving peers. Source sends one block to each participating peer for redistribution, one block in parallel to all requesting peers. Each requesting peer forwards the blocks received from the sources to the other requesting peers. After this, each peer works as a source for its own cluster. Each cluster is formed by the source S of upload capacity B_S and N_1 requesting peers R_i with an average upload capacity C_R. Each source S distributes its contents in two different routes: (1) through the content-requesting peers and (2) directly from the source. The route 2 is chosen only when the source still has upload capacity after exhausting routes 1. Thus, the distribution throughput Θ, which represents the amount of content sent to the requesting peers per second is defined as

$$\theta = \begin{cases} B_s, & B_s \leq B_R \\ B_R + \dfrac{B_S - B_R}{N_1}, & B_S \leq B_R \end{cases} \tag{4}$$

where

$$B_R = \frac{N_1}{N_1 - 1} C_R \tag{5}$$

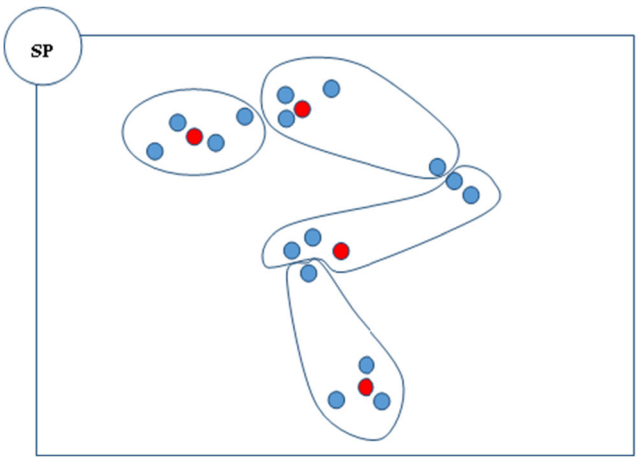

Fig. 4 Cluster size constraint

4 Implementation

This work adds scalability to the collaborative architecture presented in [11] and compares its performance with other similar architectures in the literature. To reach this objective we make use of a scalable P2P simulator called PeerSim [22]. This simulator is an extremely scalable simulation environment that supports dynamic scenarios such as churn and other failure models [22]. PeerSim has been written in Java

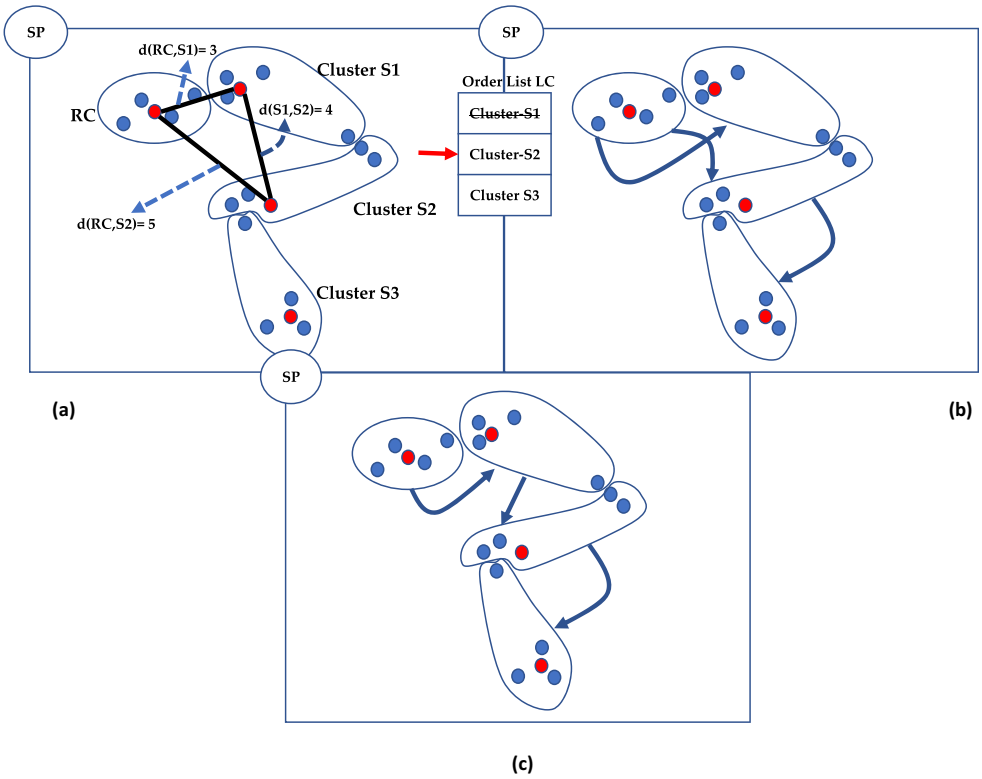

Fig. 5 Cluster size constraint. 5-ary tree creation process. (a) Initial conditions; (b) final tree for K = 1; and (c) final tree for K = 1.5

programming language, and the simulator classes can be extended to implement new peer-to-peer protocols. PeerSim consists of two simulation engines: cycle-based and event-driven. In cycle-based mode, authors claim simulation may reach 10^6 nodes. The engines are backed by many flexible components with a configuration mechanism, which can be fully configured and customized. The event-based engine is less efficient in terms of computing resources, but more realistic in its approach.

The PeerSim simulator is based on several components, which can be divided into protocols, nodes and controls. In order to improve the work environment, we use the Eclipse IDE due to its portability. To implement our collaborative architecture in PeerSim, we have developed a protocol called Hybrid Kademlia Protocol, which is a substrate between the application layer and the transport layer. Some classes from the Kademlia module [23] are taken as references and adapted in order to implement our protocol. Adapted modules of Kademlia protocol in Peersim are shown in Fig. 7.

The simulation module is customizable through a simple context using configuration files. These allow us to manipulate the parameters of all networks in order to establish the

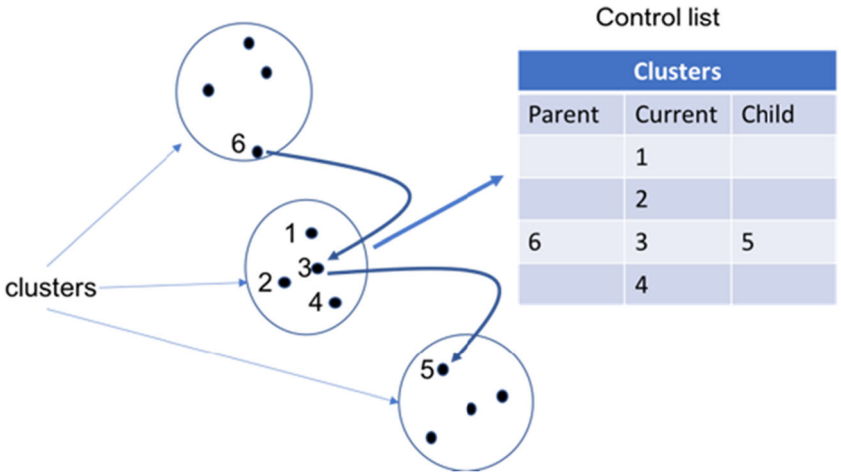

Fig. 6 An example of the control list carried out by all peers

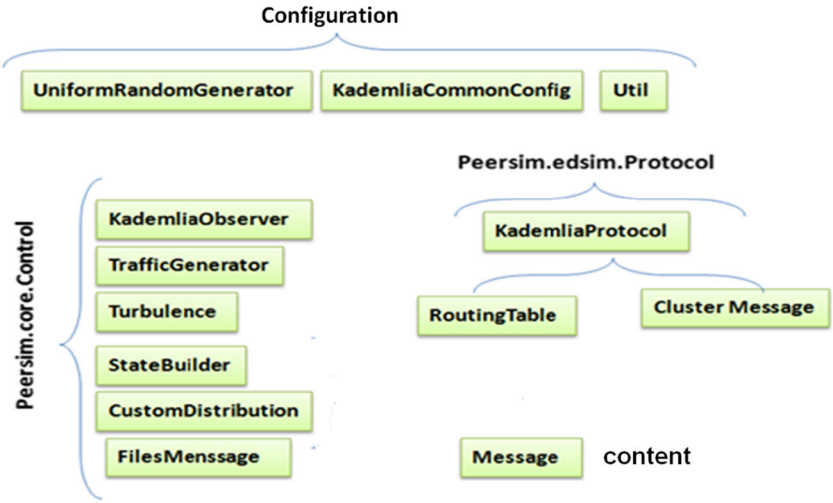

Fig. 7 Adapted modules of Kademlia protocol in PeerSim

various controls. Some of the parameters are BITS, which specifies the length of the ID, K, which is the number of calls to all replication system parameters, and ALPHA, which is the number of simultaneous search actions allowed by the protocol. The configuration file can invoke the command line to file. TXT, or can do so using the Eclipse IDE. The control protocol is very important because it allows us to simulate the dynamism and real scenarios of the nodes. These controls allow us to manipulate the traffic and the turbulence in the network, along with other events. Each control module allows for the identification of the peers who are outside the network or of the waiting time required for the distribution of content. Many of these controls have been pre-designed for the simulator. During the implementation of our architecture, a class known as Cluster class was created, which generates the group of nodes, distributes the content, and identifies the peer with its corresponding fragments. This class also allows for the manipulations of nodes within the hierarchical structure of our collaborative architecture.

Message routing plays an important role in our protocol, because when a message arrives at a node, the node can decide which route to use to send the message (a new node or the nearest node). Other classes created in our protocol are used to simulate the node's dynamicity, fragmentation of the content and the packet loss.

5 Results and discussions

We evaluate the scalability, intrinsic robustness, and cluster size of our proposed architecture based on the content distribution time to all peers using real and simulated experiments. Our first experiment, compares the performance of Mutualcast and the hierarchical collaborative multicast scheme using a small prototype over the PlanetLab Network [24]. Due to limitations of real experiments regarding the number of participant peers, we additionally simulated and evaluated the performance of our architecture in a second experiment using 500 and 1000 peers. Our third and final experiment, compares our architecture against

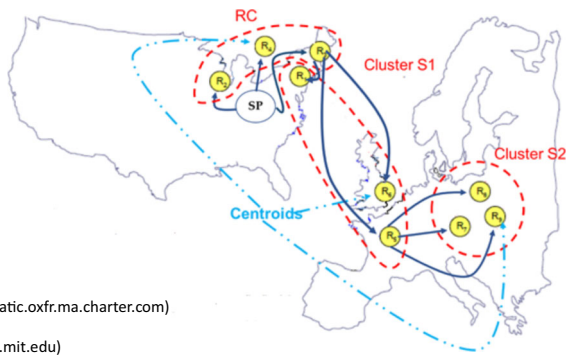

Fig. 8 PlanetLab experimental set-up and content distribution tree

PlanetLab nodes:
Source peer (SP):
SP – University of Pittsburgh (planetlab2.cs.pitt.edu)
Requesting peers:
R_1 – Worcester Polytechnic Institute (WPI): (75-130-96-13.static.oxfr.ma.charter.com)
R_2 – University of Chicago (planetlab3.cs.uchicago.edu)
R_3 – Massachussets Institute of Technology (planetlab7.csail.mit.edu)
R_4 – University of Toronto (pl2.csl.utoronto.ca)
R_5 – LIP6 – Université Pierre et Marie Curie (planetlab-01.lip6.fr)
R_6 – University College London – UCL (planetlab1.net.research.org.uk)
R_7 – Wroclaw University of Technology (planetlab1.ci.pwr.wroc.pl)
R_8 – TP-RD-Warsaw (planetlab1.warsaw.rd.tp.pl)
R_9 – Warsaw University of Technology (planetlab3.mini.pw.edu.pl)

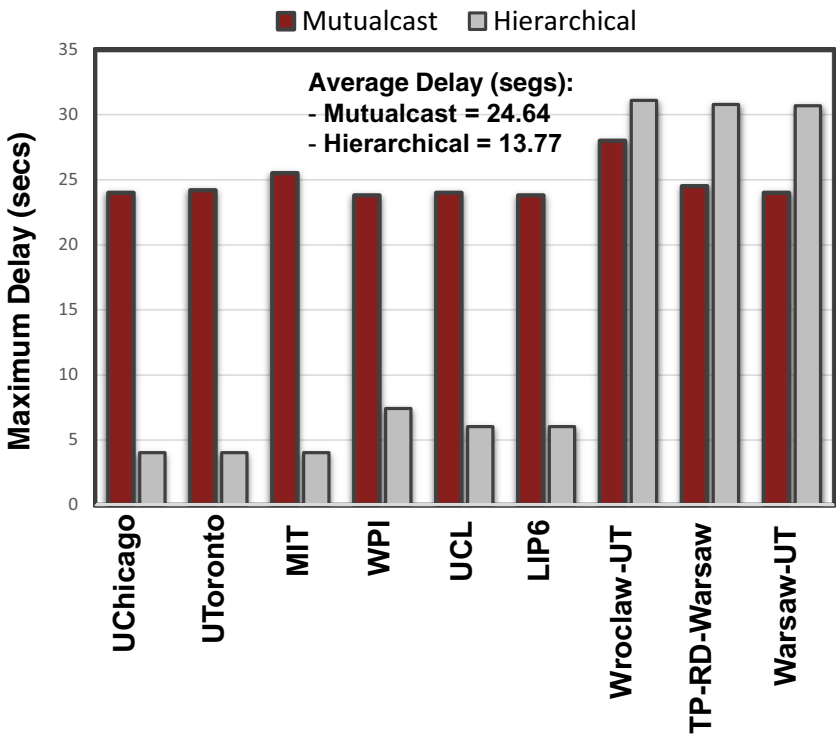

Fig. 9 Maximum delay comparison between Mutualcast and Hierarchical Collaborative [24]

two well-known P2P technologies in the literature, Super-Peer and Kademlia. In the experiments, 1.5 Mb and 5 MB files were distributed among all requesting peers.

5.1 Scalability: Hierarchical tree vs Mutualcast

A limitation of Mutulcast is related to its scalability. Although, part of our proposed architecture is inspired in Mutualcast, scalability is considerably improved by using clusters of peers organized into a unique distribution tree, that improves (in the average) the content distribution time. Scalability between our scheme and Mutualcast is compared in terms of delivery delay [24]. For this, a broadcast group of 10-peer PlanetLab [25] topology was created as depicted in Fig. 8. The SP is located at the University of Pittsburg, while requesting peers were spread out in the following academic centers: University College London (UCL), Worcester Polytechnic Institute (WPI), LIP6 (UPMC), MIT, University of Toronto,

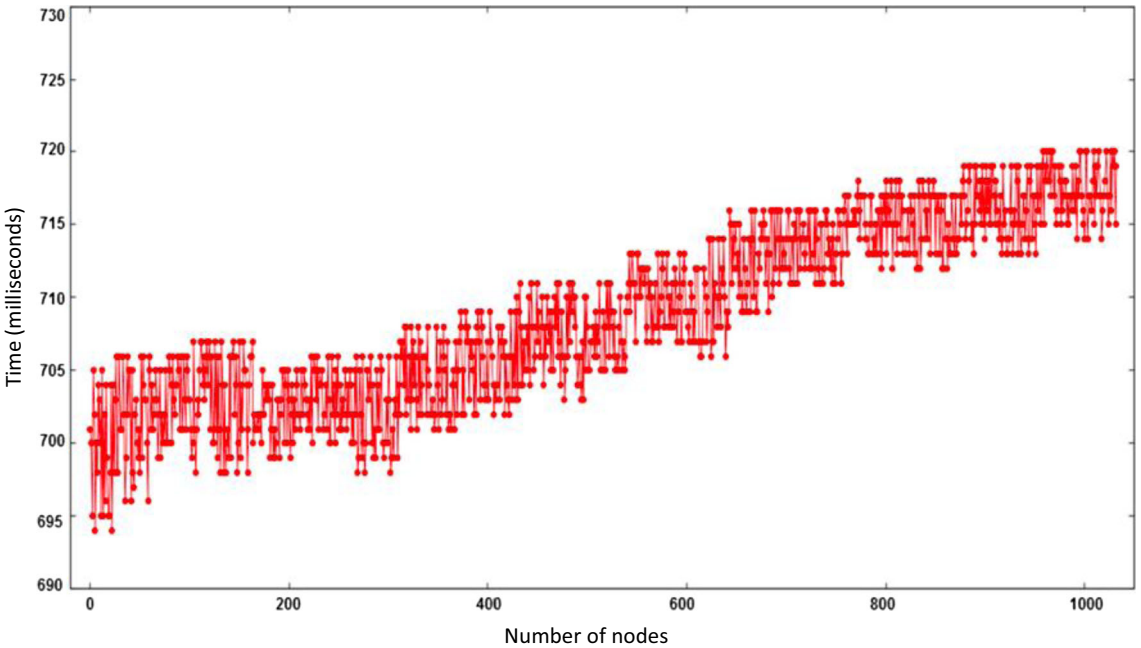

Fig. 10 PeerSim Propagation of a message in our collaborative architecture

University of Chicago, Warsaw-UT, Wroclaw-UT, and TP-RD-Warsaw. The resulting hierarchical tree after applying algorithm 1–3 (see section 3) is unary. That is, SP sends the blocks of content to closest peers R_2, R_3, and R_4 in RC; RC forwards the receiving blocks to cluster S1 (R_1, R_5, and R_6) through R_2 (the closest peer to S1 centroid), and finally, S1 forwards to S2 (R_7, R_8, and R_9) through R_5 (the closest peer to S2 centroid). During the clustering process, WPI peer was assigned to cluster S1 because of its PlanetLab connectivity was too slow despite the fact that spatially speaking it is closer to SP. Similarly, the connectivity of all nodes in Poland were also too slow, reason for which they were grouped by our clustering algorithm as the last cluster (S2).

Our Hierarchical Collaborative Topology (in Fig. 8) and Mutualcast were compared in terms of content delivery delay, in which thirty independent experiments were conducted over different days and times. The time delay in receiving the complete file content (of size 1.5 MB) at each peer was recorded,

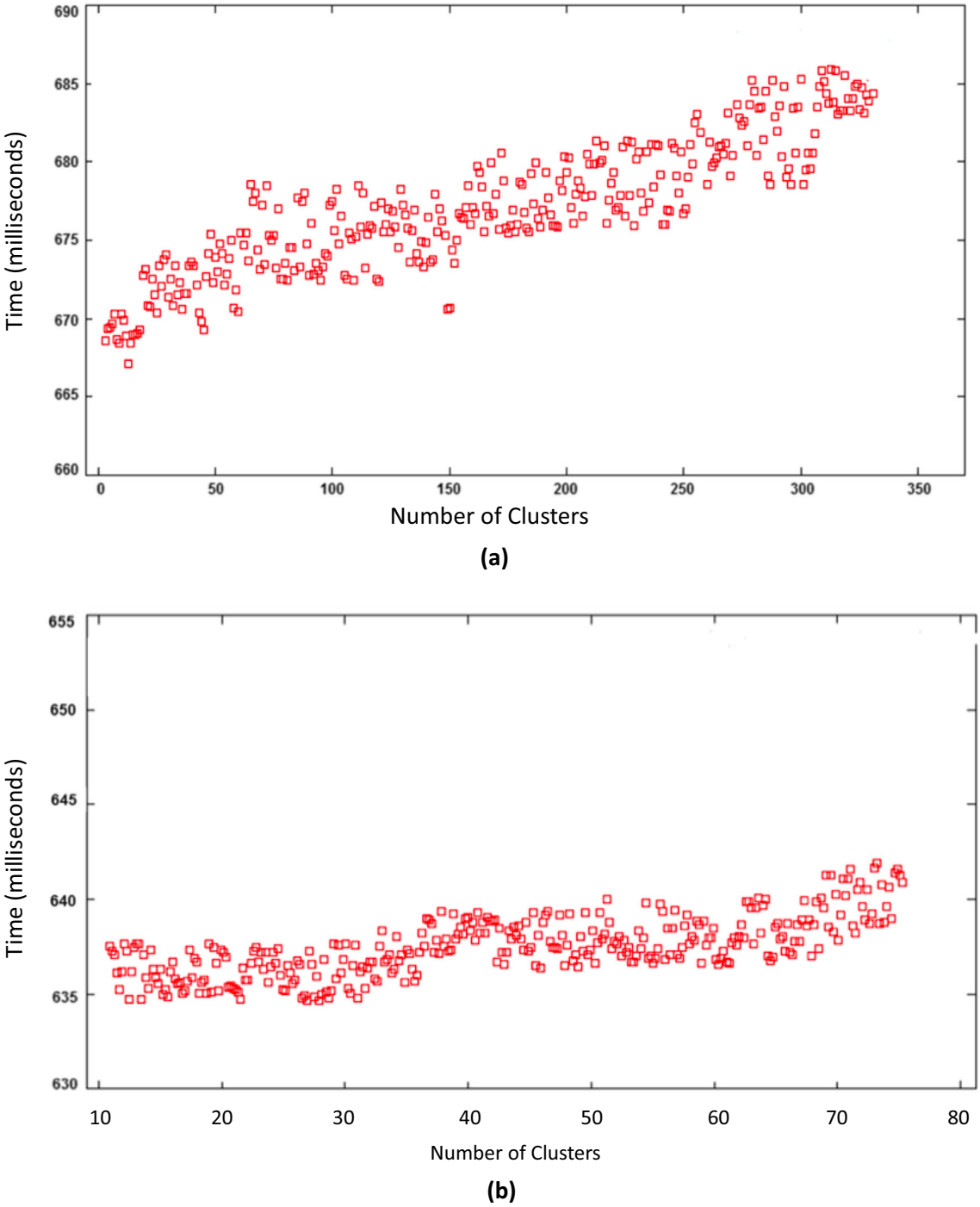

Fig. 11 Delivery delay. a) cluster size = 3 and b) cluster size = 12

and the scheme with smallest average delivery delay is assumed to present the best overall performance, as depicted in Fig. 9. When our approach is used, the average peer delivery delay is reduced by around 56% with respect to Mutualcast. This improvement is attributed to the fact that the source is close to the cluster at the highest level of the hierarchy. Thus, the throughput between the source and this subset of requesting peers is larger than the throughput between the source and the rest of the requesting peers (peers with slow connectivity are sent far down in the hierarchal tree to avoid affecting the overall content delivery delay per peer). The second fact is that using a hierarchical approach, the peers in the local clusters avoid the connection to distant peers into the overlay topology as in Mutualcast, wherein all requesting peers are fully connected. Slow peers in fully connected topologies (due to traffic, slow bandwidth, etc.), increase the average content delivery delay per peer (system is as fast as its slowest ink), as in the case of Poland, where all peers had bad connectivity.

In order to test the scalability of our scheme at higher levels with hundreds of peers, we simulated (see section 4 for details) the propagation delay of a 5 Mb file to all nodes in the *n-ary* tree network. The experimental set up consisted of $N = 1000$ peers, k = 333 clusters, cluster size $n = 3$ yielding a 10-level binary tree after applying algorithms 1–3 in section 3 (Mutualcast cannot handle this number of peers). Figure 10 shows results for this experiment, in which the first (1) and last peer (1000) in the horizontal axis are the closest and farthest to SP respectively. Our measurements of time (in milliseconds) were taken from the construction of the message until the reconstruction of the content and its defragmentation to generate a new hierarchical cluster level. Requests from peers building the cluster were intermediate operations in the protocol. Results show that the time spread of the content into the clusters increased consistently; as the tree architecture becomes deeper (increasing number of levels), intermediate and end peers require more time to regroup the total content of the transmitted media. The first five hundred peers receive the full content of the file more quickly than the rest of the peers, as expected. However, for this simulation the difference is not significant, while the first node received the complete content in 694 ms, the last node received it in 720 ms (all peers in the simulation are considered to have good connectivity and small RTTs).

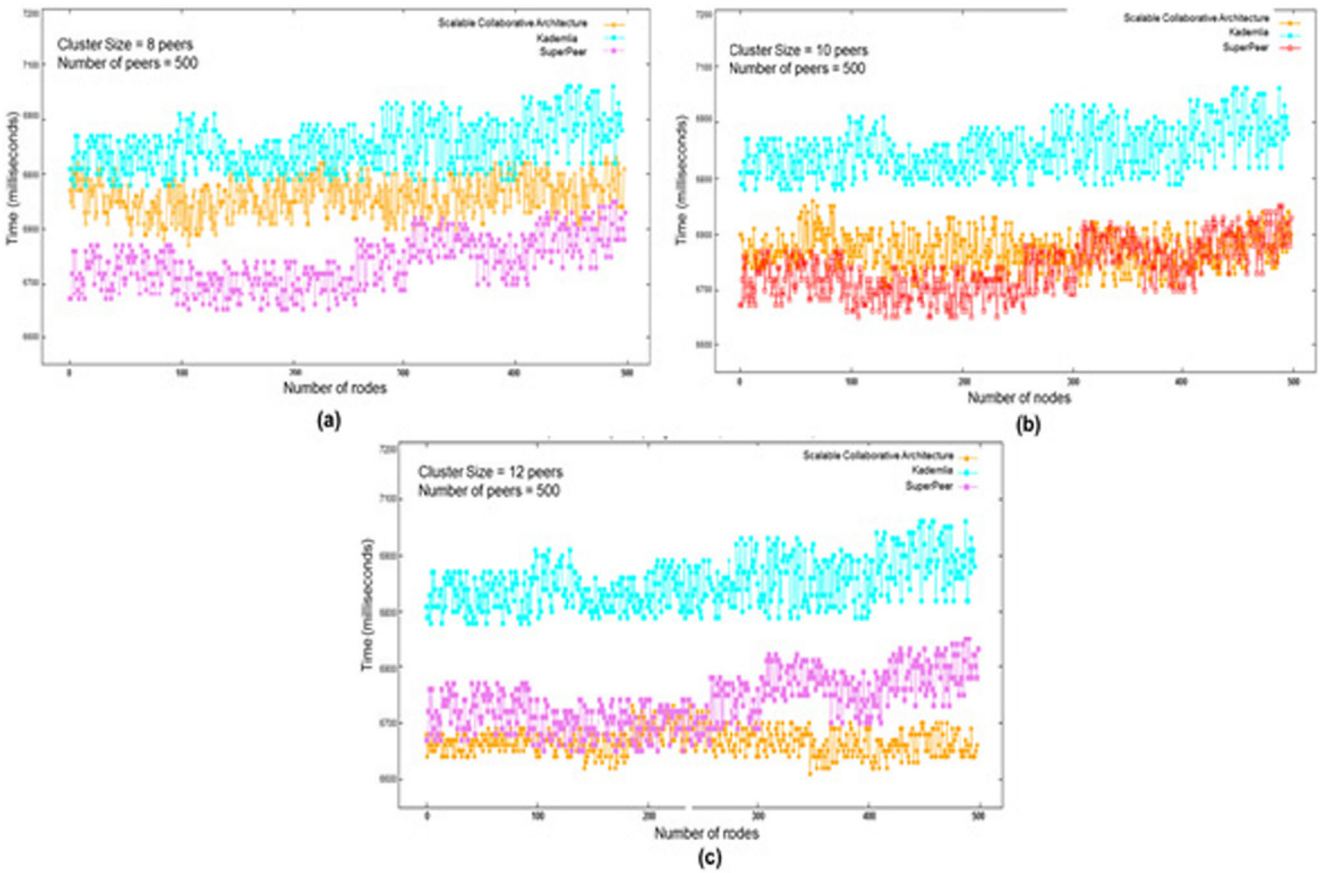

Fig. 12 Delivery delay comparison of Kademlia, Super-Peer and our Hierarchical (scalable) collaborative architecture. (a) cluster size = 8; (b) cluster size = 8; and (c) cluster size = 12

5.2 Cluster size impact

In this section, the impact of the size of the clusters during content distribution is evaluated using clusters of size 3 and 12 to distribute the content to 1000 peers, as shown in Fig. 11. These results represent the average measurements of ten simulations done with our proposed architecture. A collaborative architecture with clusters of size 3 requires a *n-ary* tree with 333 clusters to distribute the complete content to all peers, while an architecture with cluster of size 12 requires a *n-ary* tree with 42 clusters. Cluster with 3 peers commonly generates a deeper tree than cluster with 12 peers, causing a longer delay in distributing the full content to all nodes. In this case, the first peer received its complete content in 666 ms, while the maximum delay in which the last peer received its complete content was 686 ms. Clusters with size 12 had a better delivery time than the size 3 cluster; the first peer received its entire content in 634 ms, while the maximum delay to receive the complete content for all peers was 642 ms.

The results from our simulations show that the size of the cluster plays an important role in our architecture, because it introduces benefits in two important ways. The first benefit is derived from the facts that having a larger cluster means that there is a greater robustness and fault tolerance in the group. The second benefit is that the architecture gains scalability, which means that most of the requesting peers obtain the content more quickly. Traffic and turbulence factors allow us to generate a more realistic simulation of the network behavior. If a node loses communication with other nodes, it does not interfere with the reunification of the content because there are more nodes in the cluster that would provide support. For smaller clusters, a loss of content is more likely and peers have to retrieve the contents from a higher level of the distribution tree, which introduces a bigger delay. Architectures with small cluster (e.g. with 3 peers) are not robust because the clusters are very small, and if a peer left the cluster, it would be inoperative. Also, this type of architecture generates many small clusters, and the distribution tree requires many levels to organize all these clusters.

5.3 Comparison with other technologies

Finally, we compared the performance of our collaborative infrastructure with Super-Peer [12] and Kademlia [13]. We selected these protocols because of their similarities to our architecture proposed in this paper. Our simulation used a network with 500 nodes to evaluate these three architectures. In our first test, our collaborative architecture was constructed using a size 8 cluster. Figure 12a shows these results. In this case, we can see that Kademlia presents the highest distribution delay, while Super-Peer has the lowest distribution delay. The distribution delay of our architecture is between that of Kademlia and Super-Peer. Our second test considers a size 10 cluster in the collaborative architecture. The results from this experiment are shown in Fig. 12b. We can see how Kademlia continues with the same behavior, because its delivery times are high. However, delivery delay in our collaborative architecture is now very similar to the delivery delay in the Super-Peer architecture. In our last test, the cluster size in our collaborative architecture is increased to 12 peers as shown in Fig. 12c. In this case, our scalable collaborative architecture presents better performance than Kademlia and Super-Peer in terms of delivery delay, mainly in the farthest nodes (node 250 to node 500). Our results demonstrate that cluster size has an important impact on our scalable collaborative architecture, the bigger the cluster size the better the overall delivery delay. Since internally our clusters work as in Mutualcast, the cluster size is restricted to at the most 15 peers.

6 Conclusions and future work

In this work, we have developed a new hierarchical and scalable P2P architecture for fast and robust content distribution from a source to multiple nodes. In our architecture, we use time-proximity for grouping peers into clusters and clusters into a hierarchical interconnected *n-ary* tree in which, content is distributed concurrently within clusters (horizontal distribution) and among clusters in a top-bottom direction (vertical distribution). In the first place, we concentrated on evaluating critical issues in delay sensitive scalable computing systems, such as scalability (as a number of receiving peers and cluster size), robustness and delivery delay in our architecture. We found that our scheme performance (scalability and robustness) is proportional to cluster size. That is, as the number of receiving peers in a cluster increases the better the content distribution time and robustness of the system. In the second place, we compare our scheme against popular distribution schemes in the literature such as Kademlia and Super-Peer. Results show that our scheme provides a lower delivery time and better scalability, maintaining a reduced number of connections. As a future work, we are working on replacing Mutualcast content delivery in our clusters by an efficient optimized multicast scheme supporting a greater number of peers per cluster, capable of more demanding data content delivery such as video streaming.

References

1. Research Nester. Enterprise Video Conferencing Market: Global Demand Analysis Opportunity Outlook 2023. URL: http://www.researchnester.com/reports/enterprisevideo-conferencing-market-global-demand-analysis-opportunity-outlook-2023/164 (accessed on 01/05/2017)
2. Zion B. (2017) Market Research. IPTV Market for Advertising and Marketing, Media and Entertainment, Gaming, E-Commerce, Healthcare and Medical, Telecommunication It and Others - Global Industry Perspective, Comprehensive Analysis, and Forecast, 2015 2021. URL: https://www.zionmarketresearch.com/. Consulted May 1
3. Deering SE (1988) Multicast Routing in Internetwork and Extended LANs. In: Proc. of the ACM SIGCOMM, pp. 55–64, Stanford, CA, USA
4. Zhang B, Wang W, Jamin S, Massey D, Zhang L (2006) Universal IP Multicast Delivery, In: Computer Networks, Volume 50, 781–806
5. Gau V, Wu P-J, Wang Y-H, Hwang J-N (2002) Chapter1: Peer-to-Peer Streaming Systems. In: Ubiquitous Multimedia Computing edited by Q. Li, T. K. Shih. CRC Press
6. Chu Y, Rao SG, Seshan S, Zhang H (2002) A Case for End System Multicast. In: IEEE Journal on Selected Areas in Communications, Volume 20, Num. 8, pp. 1456–1471
7. Milojicic DS, Kalogeraki V, Lukose R, Nagaraja K, Pruyne J, Richard B, Rollins S, Xu Z. (2003) Peer-to-peer computing. In: Technical Report HPL-2002-57R1, HP Laboratories, Palo Alto, USA
8. Jannotti J, Gifford DK, Johnson KL, Kaashoek MF, Otoole Jr. J. W. (2000) Overcast: reliable multicasting with an overlay network. In: Proc. of the OSDI00, pp. 197–212, San Diego, CA, USA
9. Banerjee B, Bhattacharjee B, Kommareddy C (2002) Scalable Application Layer Multicast. In: Proc. of ACM SIGCOMM, pp. 205–217, Pittsburgh, PA, USA
10. Castro M, Druschel P, Kermarrec A-M, Rowstron A (2002) SCRIBE: A large-scale and decentralized application-level multicast infrastructure. In: IEEE Journal on Selected Areas in Communications,Volume 20, Num. 8, 1489–1499
11. López-Fuentes FA, Steinbach E (2007) Hierarchical Collaborative Multicast. In: Proc. of the 15th ACM Int. Conf. on Multimedia, pp. 763–766, Augsburg, Germany
12. Yang B, Garcia-Molina H. Designing a super-peer network. In: Technical Report, Stanford University. Available online: http://ilpubs.stanford.edu:8090/594/1/2003-33.pdf (accessed on 07/05/2017)
13. Maymounkov P, Mazieres D (2002) Kademlia: A Peer-to-peer Information System based on the xor Metric. In: Proc. of the Int. Workshop on Peer-to-Peer Systems, Cambridge, MA, USA
14. Wang F, Xiong Y, Liu J (2007) mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast. In: Proc. of the 27th Int. Conf. on Distributed Computing System, Toronto, Ontario, Canada, pp. 49–56
15. Magharei N, Rejaie R (2006) Understanding Mesh based Peer-to-Peer Streaming. In: Proc. of the 16th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Newport, RI, USA
16. Tran DA, Hua K, Do T. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In: IEEE INFOCOM, Mar. 2003, pp. 1283–1292 25
17. Magnetto A, Gaeta R, Grangetto M, Sereno M (2010) TURINstream: A Totally pUsh, Robust, and effIcieNt P2P Video Streaming Architecture. In: IEEE Transactions on Multimedia, vol. 12, no. 8, pp. 901–914
18. Li J, Chou PA, Zhang C (2005) Mutualcast: An Efficient Mechanism for One-To-Many Content Distribution. In: Proc. of the ACM SIGCOMM ASIA Workshop, Beijing, China
19. Celebi ME, Kingravi HA, Vela PA (2013) A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm. In: Expert Systems with Applications, 40(1): 200210
20. Hamerly G, Elkan C (2013) Learning the K in K-means: in: proc. Of the 7th annual conference on neural information processing systems
21. Zhu S, Wang D, Li T (2010) Data clustering with size constraints. In: Knowledge-Based Systems 23(8), 883889
22. Montresor A, Jelasity M (2009) Peersim: A scalable P2P simulator. In: Proc. of the Int. Conf. on Peerto-Peer Computing, pp. 99–100, Seattle, WA
23. Bonani M, Furlan DA (2010) Kademlia Module for PeerSim. In: Technical Report, University of Trento
24. López-Fuentes FA (2009) Video multicast in peer-to-peer networks. 138 pp, Verlag Dr. Hut, Munich
25. Peterson L, Roscoe T (2006) The Design Principles of PlanetLab. In: Operating Systems Review (OSR), Volume 40, Num. 1, 1116

Rogelio Hasimoto-Beltran is a researcher in Computer Sciences at the Center for Research in Mathematics-CIMAT in Guanajuato, México. He received his B.S. in Oceanology (with honors) from the University of Baja California, Mexico, in 1985 and his M.S. in Computer Science from the Center for Scientific Research and Higher Education at Ensenada (CISESE), Mexico in 1990. He received the Ph.D. degree in Computer and Electrical Engineering from the University of Delaware, USA in 2001. After his Ph.D., he spent two years at Akamai Technologies (a leader enterprise in Multimedia Content delivery) as a Senior Software Engineer. In February of 2003, Dr. Hasimoto joined the Department of Computer Sciences where he was tenure associate researcher and promoted to Full-Time researcher position. He has been visiting associate professor at the University of Illinois at Chicago (UIC) during 2009–2010. Dr. Hasimoto has published over 40 technical papers in refereed conferences and journals in the area of image processing, computer vision, and multimedia networks. His current research interest includes Robust Multimedia Communication, Error Concealment, Face Detection and Recognition, and Chaotic Encryption. E-mail: hasimoto@cimat.mx. Address: Jalisco S/N, Col. Valenciana, Guanajuato, Gto, México. 36,023

Francisco de Asís Lopez-Fuentes received the B.Sc. degree in electrical engineering from Oaxaca Institute of Technology, Oaxaca, México, in 1988. He received the M.Sc. degree in Computer Science minoring in Networking from Monterrey Institute of Technology (ITESM), Atizapán, México, in 1998. From 2003 to 2008 he was a member of the research staff of the Media Technology Group in the Institute for Communication Networks at the Technische Universität München (TUM), Munich, Germany, where he received the Engineering Doctorate in 2009. In December 2008, Dr. Lopez-Fuentes joined the Department of Information Technology of Universidad Autónoma Metropolitana-Cuajimalpa, México City, México, where is currently an associated professor-researcher for Networks and Distributed Systems. From 1996 to 2003, he was a professor-researcher in the Institute of Electronic and Computer Science at the Technological University of Mixteca, Huajuapan, México. From 1988 to 1994, he was a design engineer in the Engineering and Software Development Department at Siemens, México. His current research interests are in the area of networked and interactive multimedia systems, peer-to-peer networks, distributed systems as well as network security. E-mail: flopez@correo.cua.uam.mx. Address: Av. Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa, Delegación Cuajimalpa de Morelos, Ciudad de México, C.P. 05348.

Misael Antonio Vera-Lopez received the B.Sc. degree in information systems and technology from Universidad Autónoma Metropolitana-Cuajimalpa in 2015. During his B.Sc. studies he worked on simulation for peer-to-peer networks and video distribution.