Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2013, Article ID 312376, 11 pages
http://dx.doi.org/10.1155/2013/312376

Research Article

Peer-to-Peer Multicasting Inspired by Huffman Coding

Bartosz Polaczyk, Piotr Chołda, and Andrzej Jajszczyk

AGH University of Science and Technology, Department of Telecommunications, Al. Mickiewicza 30, 30-059 Kraków, Poland

Correspondence should be addressed to Piotr Chołda; piotr.cholda@agh.edu.pl

Received 30 January 2013; Revised 20 April 2013; Accepted 7 May 2013

Academic Editor: Zhiyong Xu

Stringent QoS requirements of video streaming are not addressed by the delay characteristics of highly dynamic peer-to-peer (P2P) networks. To solve this problem, a novel locality-aware method for choosing optimal neighbors in live streaming multicast P2P overlays is presented in this paper. To create the appropriate multicast tree topology, a round-trip-time (*RTT*) value is used as a parameter distinguishing peers capabilities. The multicast tree construction is based on the Huffman source coding algorithm. First, a centrally managed version is presented, and then an effective use of a distributed paradigm is shown. Performance evaluation results prove that the proposed approach considerably improves the overlay efficiency from the viewpoint of end-users and content providers. Moreover, the proposed technique ensures a high level of resilience against gateway-link failures and adaptively reorganizes the overlay topology in case of dynamic, transient network fluctuations.

1. Introduction

For previous few years peer-to-peer (P2P) related flows (overlay traffic) have represented a vast majority of the whole Internet traffic. Thus, P2P networking has become an important branch of today's telecommunications. It involves not only file-sharing (BitTorrent, Ares Galaxy, Gnutella), but also multimedia streaming (PPLive, SopCast), Internet telephony (Skype), anonymous routing (Tor), and many aspects of content distribution networks or cloud networking.

With the increased interest in quality-requiring applications, the problem of the quality of service (QoS) assurance is a real challenge in P2P-based streaming. Live streaming transfers based on the P2P paradigm impose strict constraints on latency between a video server and peers (end-users). Since the requirements are not fully addressed by the characteristics of P2P networking, a considerable attention has been put on the overcoming techniques. One of the most basic parameters that highly influence the quality experienced by end-users is the effective downloading speed and its stability over time. A buffering process, observed by end-users as a transient interruption of content delivery, can be minimized, or even avoided, when oscillations of the downloading speed are small. To accomplish the gapless playback, the P2P application-layer links should be set over short distances with additional care about the available capacity. This task is a nontrivial problem as P2P systems were invented to be underlay-agnostic. In practice, they have not usually recognized proximity between the candidate peers nor estimate attainable bandwidth. This situation has changed and the focus has been brought to the so-called localization-aware overlays.

Here, we elaborate on the localization-aware P2P overlays: an organization of the transmission tree is not spontaneous (following the flowing process of the peers to the overlay), but instead it uses the underlying information on the Internet paths. In effect, the video quality is improved. Our method is called "Huffmies" due to its inspiration by the Huffman source coding. The Huffman algorithm constructs a tree representing a code of the shortest average codeword length, calculated as the sum of the paths from the root to the leaves (messages). The elements of the sum are weighted by the probabilities associated with the leaves. In Huffmies, a similar mechanism is used to create a multicast streaming tree, where the root is a video server and leaves represent the peers. We aim at generating the shortest average path from the root to a leaf, while the weight of a leaf is related to a measure based on the underlay-related knowledge, namely,

round-trip-time, *RTT*. With the relationship of our algorithm to the Huffman method providing the optimal tree, we are able to show the optimality of Huffmies.

The remainder of this paper is organized as follows: Section 2 overviews the context and the related literature on the covered topic. Section 3 refreshes Huffman coding definitions and notations necessary to understand the proposed solution. Section 4, the main part of this paper, describes our novel approach in detail. Section 5 analyzes simulation results, verifying that the proposed approach is a promising alternative. Finally, the paper is summarized, and a few challenges of the proposed method are discussed.

2. Background and Previous Works

With regard to the routing topology, there are the following three types of P2P streaming systems used in practice:

(a) push: structured, based mainly on multicasting tree(s);

(b) pull: unstructured, similar to the mesh-based file-sharing BitTorrent system;

(c) hybrid: combined push-pull, based on a two-layer or two-phased mesh-tree architecture.

Currently, the majority of existing implementations and methods are related to the second type [1–4]. This stems from their flexibility and the high fault tolerance. However, we focus on the tree-based multicasting. It has also received strong research attention [5–8] due to its control potential. Additionally, by the usage of the tree-like approach, we are able to prove the bound optimality with the inspiration of the Huffman source-coding algorithm. Additionally, we study the reliability performance of our method to prove its usefulness in the case of random failures.

The challenge related to P2P networks is to find the optimal topology able to satisfy each of the three entities involved in the overlay content delivery: (a) peers, (b) the Internet service providers (ISPs), and (c) content providers. Each of them has its own requirements and prominent objectives. A peer requires the high streaming rate of the content, short start-up times, and lack of prefetching. In particular, in live video transmissions, small delays between the content source and end-users are of vital importance. While interdomain traffic load is essential to ISPs, a reduction of that load should not affect the quality experienced by customers. Finally, goals of a content provider are in-between the end-users and ISPs' needs. The content provider appreciates a high customer satisfaction (enhanced video quality) and favorable network resource utilization, as it enables a successful cooperation with operators. Our method pays to the end-user QoS satisfaction; however it reconciles also other entities' matters which are respected and considered in the simulation study.

A lot of effort has been put to improve the operation of P2P systems by taking into account the network constraints; for review see [9, 10]. Sometimes, even a strong cooperation between an overlay operator and a network carrier is assumed. The topic has been recognized as an important issue for the Internet community [11]. The research was mainly focused on file-sharing with introduction of methods like the so-called biased neighbor selection in the BitTorrent context. They rely on centrally managed devices impeding dynamic and quick decisions in case of nodes departure or network partitions. Therefore, their application prospects in video multicast networks are limited, especially when failures are considered.

A classification of approaches using locality awareness in P2P overlays can be found in [12] (a) awareness of the ISPs network, to which a peer is assigned to; (b) storage of some additional information on other peers, for example, when they serve as supernodes; (c) delay assessments; (d) geolocation information as a base for the peer node selection. Our research follows the third avenue; that is, we base on the latency measured by *RTT*. This group of approach is quite well represented. The streaming tree construction idea presented in [6] involved just a delay-aware technique to build a network friendly tree (NFT). In this scheme, an arriving peer contacts and requests a streaming flow from the node that has the smallest *RTT* among all peers in the overlay swarm. In contrast to our approach, the tree is constructed in a receiver-centric way and it does not adapt to transient network conditions. A similar concept of grouping peers into clustered trees, based on *RTT*, was presented in [7]. However, this method assumes evaluating *RTT* between an end-user and static landmarks (like DNS servers), instead of ordinary overlay peers. Our approach is based on the *RTT* between peers.

On the other hand, a proposal of Kovačević et al. [13] was based on the geolocation system, introducing zones and a two-layered, distance-aware system to accomplish strict QoS requirements. We do not assume such a sophisticated method, but base on the "application-layer `ping`," which is very simple to implement. The authors of [14] proposed to construct an optimized multicasting P2P tree, assuming that the coordination data is given in advance. Again, we decided to use a much easier method.

3. Overview of Huffman Coding

A Huffman code [15] is the minimum redundancy source code, where each message (out of N messages) is represented as a prefix-free *codeword* (a "message code"). $l(i)$, a codeword length, equal to the number of symbols in codeword i, depends on an occurrence probability $p(i)$ of the ith message generation, so that the average message length

$$\overline{L} = \sum_{i \in N} p(i)\, l(i) \tag{1}$$

is minimized. Each codeword is constructed using symbols from an alphabet of t letters (the t-ary alphabet). We skip the details of the code construction, but we remind that the ultimate idea is based on the fact that messages with higher $p(i)$ are represented with shorter codewords. A Huffman *ensemble code* (see Figure 1), that is an agreement between the source encoder and the decoder about corresponding messages and codewords, can be presented in the tree form. Thus, an ensemble code and its tree are equivalent. In such

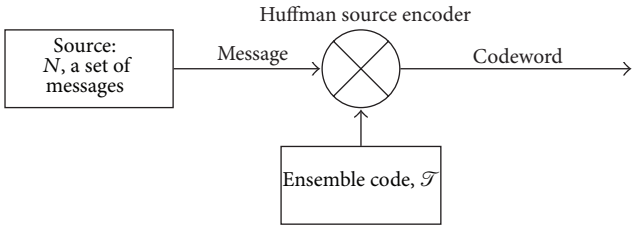


Figure 1: The Huffman coding concept.

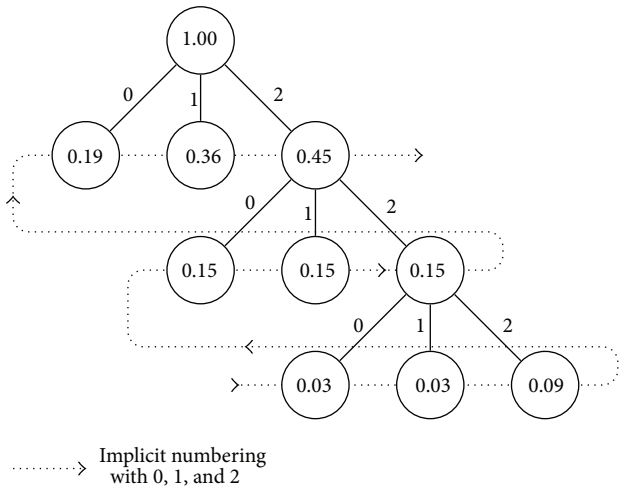


Figure 2: Example of a ternary *Huffman tree* $\mathcal{T}$. Numbers in ovals represent probabilities.

a labeled tree $\mathcal{T}$, called here a *Huffman tree* (an example is presented in Figure 2), the internal and leaf nodes have different meanings. The leaf corresponds to a single message, labeled with its occurrence probability $p(i)$. The internal node is just an auxiliary vertex with labels equal to the sum of its children's label values. In tree $\mathcal{T}$, the edges are also uniquely labeled with one of t symbols. Edge labels over a path from the tree root to a given leaf define a codeword representing the respective message. Such a definition of tree $\mathcal{T}$ is used also in [16, 17] to create an adaptive Huffman algorithm. The original static (two-pass) and newer adaptive (one-pass) Huffman algorithms differ in the ensemble code construction method and its exchange between the encoder and decoder. However, both ensemble codes have the same structure and the $\overline{L}$ value. Therefore, for the sake of interpretation purposes, we can use the static and adaptive algorithms interchangeably, since results for those approaches are the same.

4. Huffmies Design

Below, we present "Huffmies," the centralized and distributed approaches that create a streaming topology for a multicast tree-enabled P2P overlay network. Both approaches are inspired by the Huffman algorithm. The summary of the notation related to the Huffman coding and Huffmies is given in Table 1.

4.1. Centralized Huffmies. Each P2P streaming network uses a content server s, which can upload to only a limited number of nodes simultaneously due to the bandwidth and computing resource limits. Among all the active peers subscribed to the stream, the server has to choose at most t peers that will receive the data directly from it. The centralized Huffmies algorithm uses a single device managing all connections between the peer nodes in the entire overlay swarm. For the rest of this subsection we assume that server s, which is a reliable, rich in resources, and credible node, can take responsibility of the overlay network construction and control. Hence, the terms "centralized device" and "server" will be used interchangeably here.

The Huffmies method introduces $p(i)$ value, namely, *peer i aggravation*. It forms a concise representation of peer i's network resources in such a way that higher values indicate inferior resources due to a tight bandwidth limit or high latency of the network access. Finding strict and accurate value of $p(i)$ is not a trivial problem, since it is influenced by many unpredictable parameters (the location, available capacity, congestion). We simply use the round-trip-time metric (RTT) between a peer and the server or between a peer and another peer. The high value suggests that the given peer has considerably limited network resources or can be located in a long distance from the source node (or another peer). Nonetheless, more complex and sophisticated metrics can be involved here as well. Furthermore, as a peer receives data using a multiple-hop path, the sum of all $p(i)$ values over this path emphasizes *service aggravation* of the connection.

In the centralized scheme, only the streaming server gathers all individual $p(i)$ values, independently measured by the peers and sent to the server. The final value of $p(i)$ is defined as an aggregated sum of RTTs between peer i and all other peers in the swarm, including the server:

$$p(i) = \sum_{j \in (N \setminus \{i\}) \cup \{s\}} RTT_{ij}, \tag{2}$$

where RTT_{ij} represents the evaluated RTT to node j observed from the peer i's point of view (we set $p(s) \equiv 0$). All probing packets are analyzed individually by each node in a swarm and the computed result of $p(i)$ is forwarded to the server. As the central device has the entire $p(i)$ map, it builds a *Huffman tree*, analogously to the Huffman coding with Vitter's algorithm Λ extension [18], in a manner that $p(i)$ is treated like a probability of message i generation. In a *transmission tree* T we are constructing, a codeword length $l(i)$ is defined as a path length from the root (the streaming server s) to leaf i:

$$l(i) = d_{\mathcal{T}}(s, i). \tag{3}$$

Let us remind that the *Huffman tree* $\mathcal{T}$ is a form, where only leaves represent something real (i.e., the messages). Therefore, in order to transform a *Huffman tree* to a *transmission tree* T, a process called *climbing* is introduced (see Figure 3). It consists in using the internal vertices as something of real meaning, that is, as the peers that are indeed present in the multicast tree to both download the data and simultaneously upload it to other peers. The climbing starts from the most

Table 1: The relation of the basic notions for the Huffman coding and the Huffmies concept.

Not.	Huffman	Huffmies
s	Root node	Streaming server
$\mathcal{T}$	Ensemble code (Huffman tree)	—
T	—	Transmission tree
i	Message i	Peer i
N	Set of all messages	Set of all peers in the overlay swarm
$p(i)$	Probability of message i generation	Probability of peer i being a bad relay (peer i aggravation)
t	Number of the source alphabet symbols	Maximum number of simultaneous upload connections (slots)
—	Codeword	A route from the root server s to a leaf in a transmission tree
$l(i)$	Length of a codeword representing message i	Peer i level in a transmission tree, $l(i) = d_{\mathcal{T}}(s, i)$ (hop distance between s and i in the tree)
—	Leaf: a message; internal vertex: auxiliary	Leaf: a downloading peer; internal vertex: a downloading and uploading peer
O_i	—	Peers downloading from peer i, the offspring of i

bottom layer where each internal node, noted as "X" in Figures 3(b)-3(c), is replaced by the node with the lowest $p(i)$ value beneath this auxiliary node. While going upward, it can happen that the minimum value of $p(j)$ characterizes a node j that already has downstream nodes (a node with $p(j) = 1$ in Figure 3(c)). In such a case, peer j's child with a minimum peer *aggravation* has to take its place (a peer with $p(i) = 2$ in Figures 3(c)-3(d)) and the node j is free to jump into an upper layer. Such a process creates a *transmission tree* T with a bounding number of children equal to t. Hence, parameter t is considered as a maximum number of simultaneous unicast connections that a given peer can serve simultaneously. To simplify, we assume the same value for all peers. However, a dynamic adjustment of this value, for instance depending on the swarm size, streaming ratio, or peer's resources, is also possible.

4.2. Optimality of the Centralized Huffmies. In general, P2P tree-based streaming dissemination has to cope with conflicting objectives to design either a narrow or a short transmission tree. A wide tree, where each peer has to send data to numerous neighbors, divides the peer's upload capacity among all children. This can inflict the degradation of quality of experience (QoE), that is, the perceived quality, for the downstream peers. On the other hand, a high tree involves long paths between the streaming source and the peers. Such a tree should be averted since it imposes additional delays and decreases the system reliability in case of an intermediate node fault or departure. Thus, a convenient trade-off between the width and height should be found. The Huffmies method involves optimization of the worst-case *peer aggravation* suffered by an entire overlay swarm. We prove this fact below.

In general, each peer that arises on a path from the streaming server to a single destination node introduces bandwidth and delay limits that inflict a higher probability of service quality degradation for downstream nodes. Sum of all intermediate nodes' $p(i)$ on a path from the server to a destination node is called a *service aggravation* $s(i)$. We model a deteriorated QoE caused by a multihop overlay path and observed by peer i, as $s(i)$, where

$$s(i) \stackrel{\text{def}}{=} p(i) + \sum_{j \in d_T(s,i)} p(j), \tag{4}$$

where $d_T(s, i)$ is a path from the server to the destination peer i in a *transmission tree* T. The main goal of the Huffmies method is to minimize this parameter for the entire overlay swarm.

We can note the two following fundamental facts.

Theorem 1. *The Huffmies algorithm presented in Section 4.1 ensures that each peer receives the data from a node that has a smaller or equal peer aggravation $p(i)$ value.*

Proof. We show it by a contradiction. Assuming that peer k receives data from node j such that $p(j) > p(k)$, leads to a contradictory statement, since peer k, as a peer with smaller $p(k)$ value among other siblings, would be elected for *climbing* instead of j. □

Theorem 2. *An upper bound for service aggravation $s(i)$ suffered by peer i in the centralized Huffmies is $p(i)l(i)$.*

Proof. From the fact that the *climbing* process performed on the Huffman tree $\mathcal{T}$, as described in Section 4.1, only uplifts some nodes into higher layers, then $l(i)$ is an upper bound for the length of the transmission path from the server to the peer i, $d_T(s, i)$. On the basis of Theorem 1 we also state that $p(i)$ bounds the maximum *peer aggravation* of nodes that can occur over this path. That provides the upper bound $s(i) \leq p(i)l(i)$. □

According to the Huffman coding algorithm, which minimizes average message length $\overline{L}$ (given in (1)), and Theorem 2 we can see that the centralized Huffmies algorithm minimizes

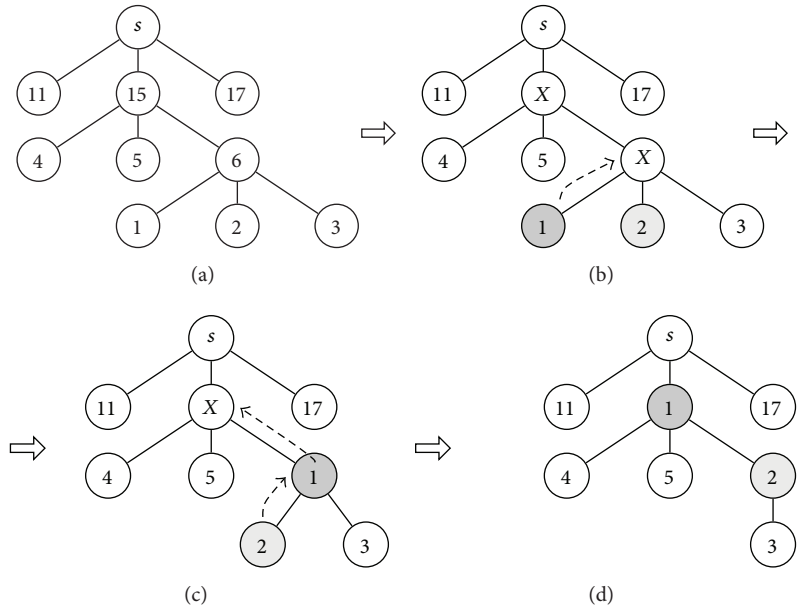


Figure 3: A *climbing* process: evolution from a *Huffman tree* (a) to a *transmission tree* (d) with two intermediate steps: (b) and (c). $t = 3$, numbers labeling vertices denote peers' aggravation $p(i)$.

the *service aggravation* bound for all nodes in the overlay swarm.

4.3. Distributed Huffmies. Application of Huffmies in a huge swarm inflicts a `ping` packet flood, since each peer has to evaluate *RTT* between all other network members. Indeed, a number of such probes at the order of $\mathcal{O}(N^2)$ limit the practicality of the centralized scheme. Overview of the consecutive stages in the distributed Huffmies algorithm is presented in Figure 4. The principal difference between the distributed and centralized version consists in a subjective estimation of the $p(i)$ value accomplished by a single node in a P2P network. In the first step, the server estimates *RTT* to each member in a swarm, which is assumed as its $p(i)$. Then, on the basis of peers' $p(i)$ values, the server forms t disjoint groups of peers. Among all the members of each group, the server elects a single peer to which it will upload the stream directly. Such a peer is called the group *leader* and it has to replicate the data towards the rest of the group nodes called an *offspring* of a given *leader*. The decision criteria used by the *leaders* to disseminate the data are completely independent and the server does not have any influence or authority regarding this issue. Practically, a *leader* performs the same steps as those of the server in the first step; that is, it selects the best *leaders* among its *offspring* and assigns them a next-level *offspring* (i.e., offspring of those nodes). A *leader* bases its decision on independently obtained $p(i)$ values. In the next steps, such a procedure is consecutively repeated until a pool of all peers in the overlay network is exhausted.

The key point, where the Huffman algorithm is engaged, concerns the selection of the *leader* nodes and assigning them appropriate peers to disseminate data downstream. To accomplish this task, each *leader* k assigns a codeword for all of peers of its *offspring* O_k, analogously to the centralized approach. Thus, it is necessary to calculate the Huffman tree in each group. Note that in this case, contrary to the centralized Huffmies, $p(i)$ is assessed only from a group leader (or the server) to each peer, but not between all pairs of peers. The corresponding probabilities $p(i)$, $i \in O_k$, are equal to *RTT* evaluated over the individual probe between the *leader* k and peer i. After applying a codeword for each peer in O_k, all peers whose first digit is the same constitute one group, obviously disjoint with other groups. Among all members of a group, a peer with the lowest value $p(i)$ is selected as a group *leader*. Now, this leader is fully responsible for dissemination of data towards other members of its group. Looking at this issue from the *Huffman tree* viewpoint, a group is constituted by all peers, whose corresponding leaves belong to the same root's subtree. Figure 5 presents two *Huffman trees* (with omitted edge labels) for $t = 2$ (binary tree) and $t = 3$ (ternary tree) with group borders for the same set of peers and the same *RTT* measurements.

Such a distributed algorithm significantly diminishes aggregated number of links that are engaged in the *RTT* probing process. While the streaming server has to evaluate all N `pings` to all nodes in the swarm, other peers in lower *transmission tree* layers make a decision about much smaller groups, whose cardinality deteriorates at the order of $\mathcal{O}(t^{-l_i})$, where l_i is a peer's layer index in a *transmission tree*.

The Huffmies method clusters peers into groups, so that the sum of members' *aggravation* is almost equal for all groups. That means that peers with large values of $p(i)$ will be present in groups of smaller number of members. This is a very advantageous and practical property. Let us remind that a low $p(i)$ value for a given peer means that it inflicts a small negative effect on other children. We argue that it is highly recommended to create groups with a comparable sum of *aggravation* indicators. For instance, one group consisting

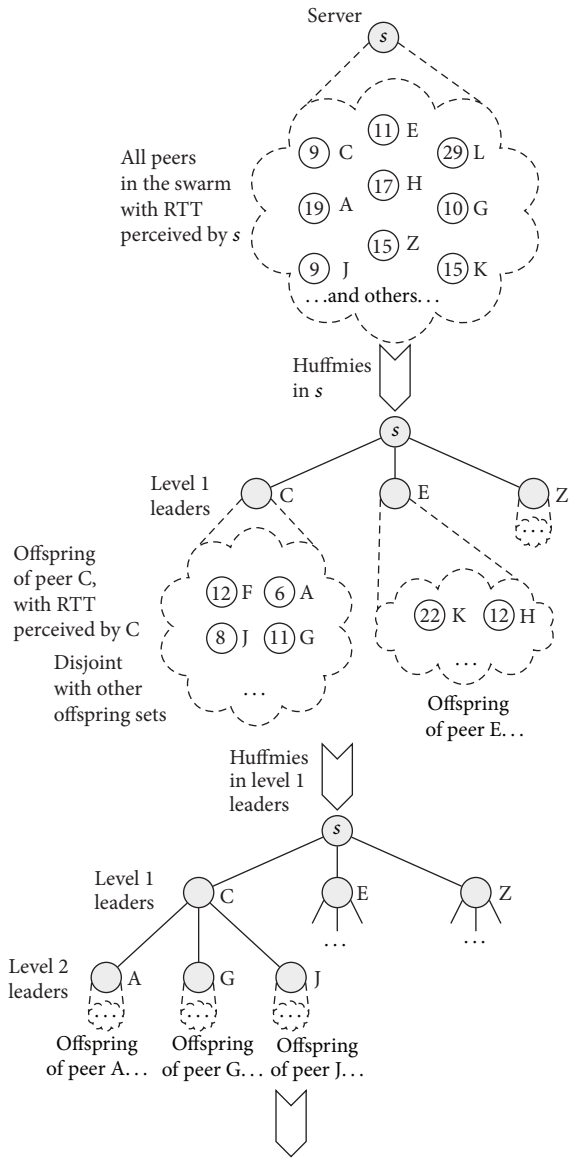


Figure 4: Overview of the consecutive stages in the fully distributed Huffmies approach. Values in circles denote $p(i)$.

of peers that are fast, reliable, and localized in a short distance from the source (all those parameters result in low $p(i)$) can aggregate a higher number of peers and the transmission quality will not be reduced. On the contrary, a high $p(i)$ value of a member in a group prevents from accepting too many other peers. It prevents from low QoS parameters (like high delay and low streaming rate), experienced by many peers, when peer i (with poor resources) becomes a relay point. In particular, when a peer has a notably higher $p(i)$ (with comparison to the values of $p(i)$ for other peers), caused by low-quality transmission (e.g., wireless access), it will constitute an independent group receiving data directly from the server to alleviate the negative effects related to this peer's poor resources.

4.4. Churn Effect. The considerations presented above assume the static environment: all peers are permanently on-line. However, in a real overlay, peers can freely join and depart the P2P network. In result, the population of the swarm and the connections within it often change, what is known as the *churn* effect. To avoid frequent chaotic rearrangements of connections in case of every peer joining or departure, we assume that the overlay members apply the Huffmies algorithm only in a specified point of time, called *reorganization*. In the interim, every new peer is temporarily serviced by a random node. On the other hand, for each *leader* of a group, a *backup leader* is defined. It is selected as a peer with the second lowest $p(i)$ value in the group. The *backup leader* takes care of a group in case of the *leader* departure. Therefore, a single peer departure does not destroy the existing multicast tree.

5. Numerical Results: Simulations

To evaluate benefits of the Huffmies concept in a video streaming network, we carried out simulations, obtained using a C# simulator developed by us. Two scenarios for performance investigation and one scenario for fault resilience were examined. For all simulations, a core network has been randomly constructed according to the model presented in [19], with symmetric 100 Mbit/s links between selected routers. Peers were grouped into five ISP networks (ISP A–E), where each ISP was connected to one of the backbone routers with a symmetric speed of 10 Mbit/s. To mimic a real-world environment, a number of customers and their access bandwidths were generated taking into account statistics offered by a connection speed estimation web service (http://www.netmeter.eu/). An interdomain network topology contains nine domains connected via fourteen links. A network topology, used for simulations, is presented in Figure 6. The content server (located in the biggest ISP A) had 2 Mbit/s uplink and 4 Mbit/s downlink bitrates. Moreover, each peer could handle at most four simultaneous upload connections ($t = 4$). To limit frequent reorganization of the network transmission tree, due to the churn effect, Huffmies *reorganization*s were separated with 5 minutes long space. All results are presented with 95% confidence intervals.

To investigate the overlay streaming efficiency and exhibit the Huffmies features, we exploited various overlay creation concepts, such as

(i) Huffmies (H),

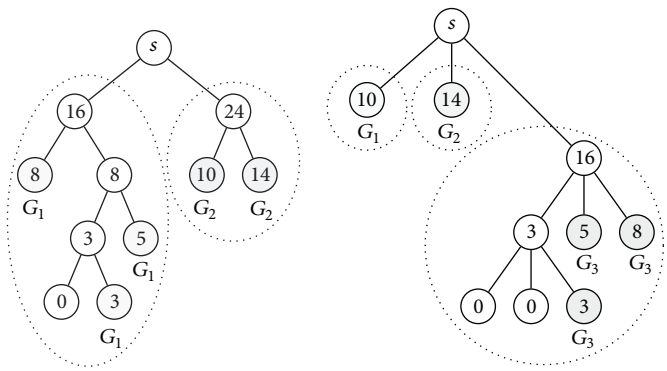
(ii) centralized Huffmies (H-C),

(iii) random (R),

(iv) Network Friendly Tree (NFT),

(v) minimum *RTT* tree (M).

The distributed Huffmies algorithm, described in Section 4.3, is denoted as the Huffmies (H) concept to distinguish it from the centralized Huffmies (H-C). The simplest random (R) method assumes that an entering peer connects to one of the randomly selected peers. It also pays attention to the maximum number of the upload connections to be handled.



(a) Binary tree (b) Ternary tree

Figure 5: Example of binary and ternary *Huffman trees*. Labels G_i given beneath nodes indicate to which group a given peer should belong. The dotted circles indicate group borders. The values in circles denote $p(i)$.

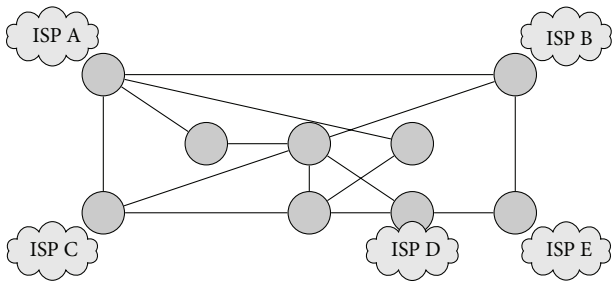


Figure 6: The interdomain network topology used in the simulations.

t. The R concept can be treated as a reference idea due to its underlay agnostic principle. In the NFT, the receiver-centric algorithm chooses a neighbor achieving the minimum value of *RTT* across all peers in the swarm. The minimum tree (M) is similar to the Huffmies concept; however, it differs with the grouping scheme: the peers sorted according to $p(i)$ in the decreasing order are consecutively pooled (with the round-robin technique) into t groups. Similarly to the Huffmies, a peer with the lowest $p(i)$ value in a group is selected as a group *leader*. The data is always transferred to t peers with the lowest *RTT* value. Figure 7 gives examples with (i) a *Huffman tree* and (ii) a minimum tree for the same *RTT* values.

We analyze four types of relative metrics which span requirements of all entities involved in the P2P transfer:

(i) *speed*: average end-user download speed,

(ii) *delay*: average delay of the content playback,

(iii) *network efficiency*: the ratio between the total load received by all peers to the total inter-ISP traffic,

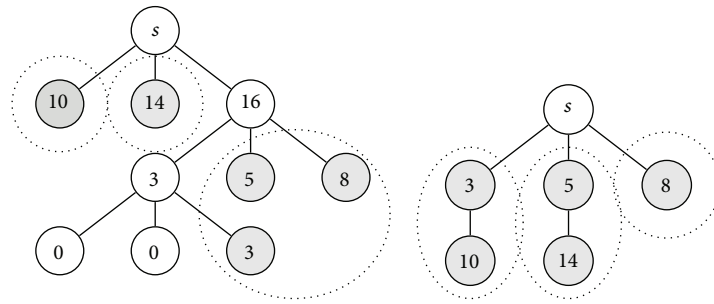
(iv) *fault immunity*: the percentage volume of peers that suffer data loss in case of a failure of a gateway (intradomain) ISP link.

All the aforementioned parameters, except for the *network efficiency*, are relevant for an end-user, since they influence QoE. The *network efficiency* shows how efficient the overlay network is in utilizing expensive inter-ISP links and, therefore, this factor is related to the ISP objective. The last interested player, a content/streaming provider, wishes to provide a streaming service with a high reliability and high quality for an end-user, while optimal utilization of network devices is also prominent. Therefore, the content provider is interested in all the studied metrics.

5.1. Scenario I: Basic Performance Evaluation. In Scenario I, we compared the Huffmies approaches, both centralized and distributed, with NFT and underlay oblivious R concepts. Live streaming content uses 256 kbit/s rate. Peers' lifetimes follow a normal distribution with mean equal to 30 and standard deviation equal to 10 minutes, while their offline times follow a normal distribution with mean equal to 10 and standard deviation equal to 3 minutes. Each simulation took 90 minutes and was repeated 24 times to accomplish credible and representative results. The warm-up period, determined according to a standard method presented by Tyszer [20, Chapter 7.3.1], oscillated in vicinity of 30 minutes.

Figure 8 presents the average download *speed* with various numbers of peers in a swarm. The centralized Huffmies (H-C) achieves the best result for small swarms and with increasing number of peers the downloading *speed* drastically deteriorates. Most likely it is caused by ping packets flood in case of a new peer joining, which has to estimate *RTT*s to all on-line nodes in a swarm. The churn effect does not inflict such degradation in case of the distributed Huffmies (H) method. Its results are slightly worse for small swarms; nonetheless it copes with scaling problems and the churn effect much better than the centralized algorithm. The NFT and R concepts achieve lower downloading *speed*s for all inspected swarms. It is also interesting to note that the R approach also suffers from the churn effect in a bigger swarm, like the centralized Huffmies (H-C). Figure 9 shows the *speed* fluctuations. The large disadvantage of NFT can be especially observed with this plot.

The *network efficiency* results are plotted in Figure 10, where NFT significantly outperforms other considered schemes. This is caused by a high clusterization of peers within each ISP network as a consequence of a mere delay-aware neighbor selection process. Other approaches obtain



(a) Huffman tree (b) Minimum tree

Figure 7: Huffman and Minimum trees for the same peers with selected *RTT*s (values 3, 5, 8, 10, and 14). Members of each group are placed within dotted circles. $t = 3$.

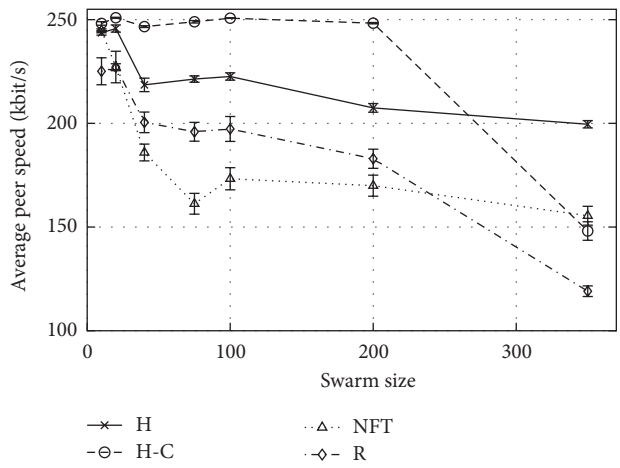


Figure 8: Average peer *speed* in Scenario I.

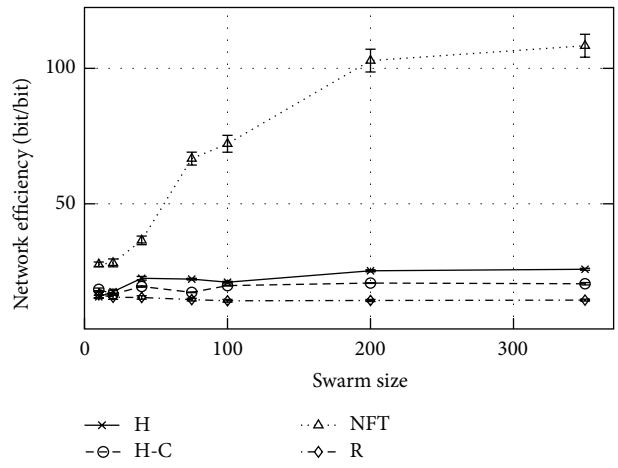


Figure 10: Average *network efficiency* in Scenario I.

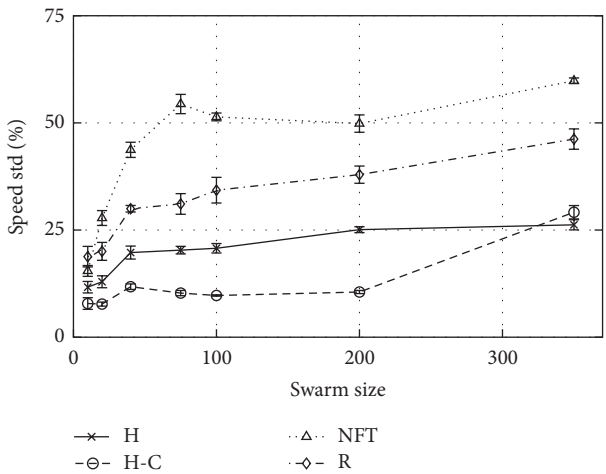


Figure 9: Average peer *speed* fluctuations in Scenario I.

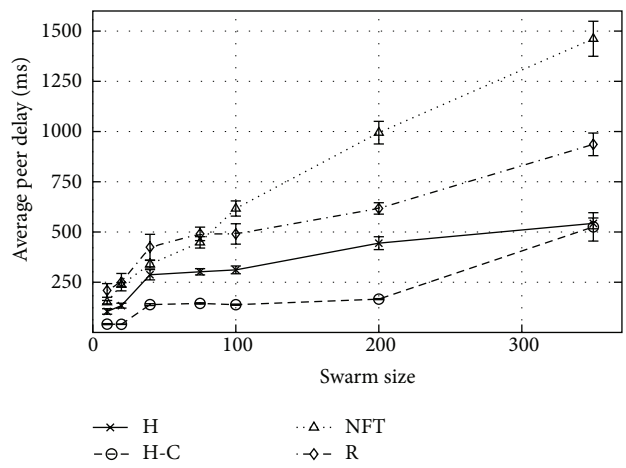


Figure 11: Average peer *delay* in Scenario I.

stable results along various swarm sizes. Altogether, both Huffmies algorithms inflict a lower inter-ISP load than a simple R approach. The *network efficiency* for the distributed Huffmies is almost doubled in comparison to the R case. The *delay* between the server and end-users is relevant for multimedia streaming, in particular when live content is involved. Figure 11 shows that the Huffmies approach keeps low delay for various swarm sizes, contrary to other concepts that notably increase the content lag as a swarm size increases. A considerable growth of the *delay* for NFT, which we observe

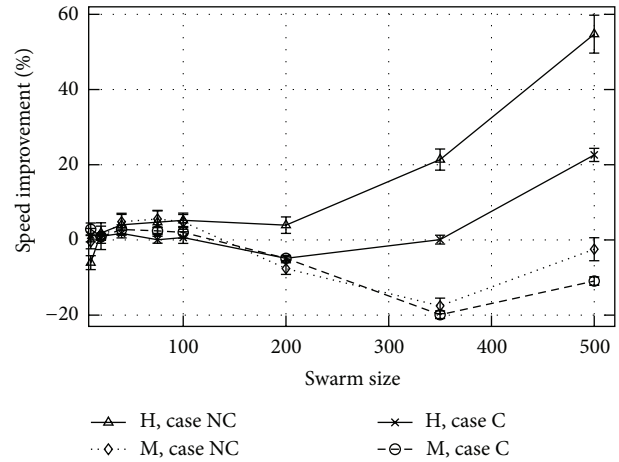
Table 2: Parameters of the performance evaluation cases in Scenario II.

Case	Switch on time	Lifetime	Offline time
NC (no churn)	0–20 minutes	Infinite	—
C (churn)	0–20 minutes	0–120 minutes	0–20 minutes

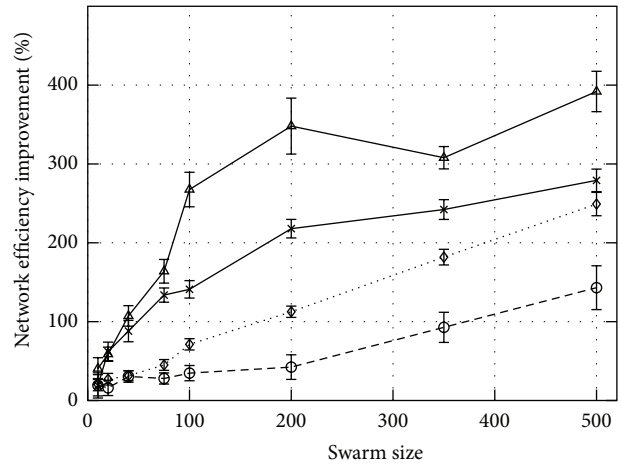
between networks with 100 and 350 nodes (although in the case of file-sharing, such a size is very large, for live streaming such an extension of the peer set is quite real or even quite small, e.g., football league match), is caused by a long average path from the server to a peer. For other concepts, a mean number of overlay hops is smaller than three, but NFT creates a topology where the data has to be relayed to reach a destination by more than five other peers on average. Every overlay hop introduces additional delay, that is, why NFT achieves high values. Based on the previous results we noticed that the centralized Huffmies provides better results for small swarms. However when the number of nodes in the overlay exceeds a threshold level, its efficiency deteriorates drastically. On the other hand, NFT gracefully utilizes inter-ISP links, but it introduces a huge delay between the server and peers and this method achieves low streaming rates. Thus, for Scenario II investigating head-to-head comparison with the reference R approach and involving bigger swarms, we decided to omit the centralized Huffmies (H-C) and NFT schemes as evidently inferior.

5.2. Scenario II: Extensive Performance Evaluation. In Scenario II, each simulation took 5 hours. The streaming content rate was equal to 128 kbit/s. To explore the churn effect impact, we exploited two cases. The first case dealt with all peers joining the overlay according to the uniform distribution during first 20 minutes and are kept on until the end of the simulation (no departures). In the second case, the peer's lifetime and offline time were obtained from a uniform distribution between 0 to 120 minutes and 0 to 20 minutes, respectively. The parameters of those cases are presented in Table 2.

An improvement, against the reference R concept, of the *speed* (Figure 12(a)) and the *network efficiency* (Figure 12(b)) is presented. We can see that the both *speed* and *network efficiency* results are better for the Huffmies approach (compared to the R and M concepts). The difference is greater when the swarm becomes larger, due to the fact that the Huffmies approach serves more traffic inside one operator domain; thus inter-ISP links (which become bottlenecks) are not heavily occupied. From Figure 13, where the *speed* for all approaches is presented, we can observe that for swarms larger than 200 peers average *speed* significantly decreases for the R and M methods. At the same time, the Huffmies approach still achieves a very good transfer speed. A relatively stable streaming rate among various swarm sizes proves that the distributed Huffmies has good scalability abilities since a bigger swarm does not result in the downloading speed degradation. Although the minimum (M) alternative is characterized by a higher network efficiency in comparison to the reference R scheme (Figure 12(b)), the average *speed*



(a) *Speed* change



(b) *Network efficiency* change

Figure 12: Scenario II: improvement against the reference R tree creation scheme.

received by peers is often lower than that in the case of the simple R method (Figure 12(a)).

One of the main challenges the Huffmies concept still has to cope with its resilience against the churn effect. Figure 12 illustrates that the improvement of both the *speed* and *network efficiency* decreases when peers join or depart the overlay network at will (case C). Note that in spite of that decrease, the Huffmies approach immensely outperforms other methods from the viewpoint of those two metrics.

5.3. Scenario III: Fault Resilience. The Huffmies approach is characterized by a decreased number of backbone links involved in the transmission. Keep in mind that the usage of additional network nodes (i.e., routers in the IP multicast applications) when the data unnecessarily traverses several times over the same link leads to single points of failure that can noticeably affect the system reliability. Here, in Scenario III, we also investigate the number of nodes that

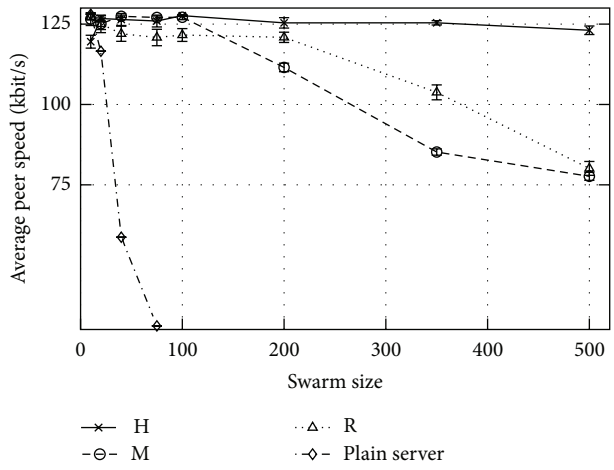


Figure 13: Scenario II: average peer *speed* depending on a swarm size.

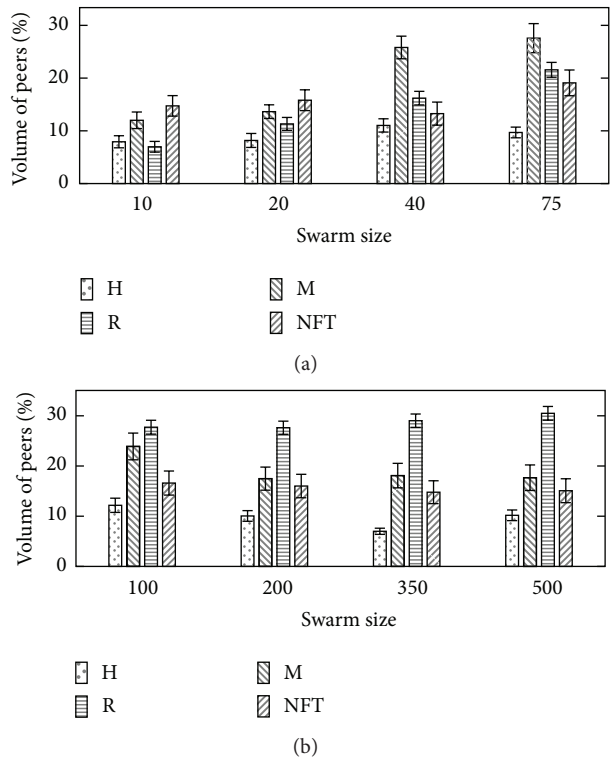


Figure 14: Peers suffering data loss in case of an ISP gateway-link fault.

suffer data loss when a (randomly selected) ISP gateway-link fault occurs. The results presented in Figure 14 are obtained from simulations, where suddenly all peers from one of the five ISP networks are pulled apart from the rest of the overlay network, resulting in the tree decay. To prevent the overlay network from the content server isolation (in case of ISP A fault), the server is located in a separate network which permanently services the content streaming. In case of the Huffmies algorithm, the percentage of suffering peers was usually 10–20% less than that in the M and R approaches, respectively. We envisioned that the NFT concept, characterized with a good clusterization observed in Scenario I, will achieve low number of decoupled nodes in case of an ISP break. However, it achieved values higher by only 5–10% than the proposed approach and oscillated in vicinity of 15%. The reason for it can arise from a long average path from the server to a peer as we observed in Section 5.1. The accomplished results show that when using the Huffmies method, connections are densely and efficiently clustered within ISP networks. Thus, isolating one region from the entire community has a smaller impact on the rest of overlay nodes, proving that network resilience against ISP gateway-link faults, using Huffmies, is higher than that in the other considered concepts.

6. Summary

We present a novel method, "Huffmies," to be used in P2P video live streaming networks. It leverages the Huffman coding method to create a connection graph fitted to the underlay network properties and peers heterogeneous network access. First we show the optimal centralized algorithm, difficult to implement in a real world due to the scalability problems and intricate probing process. Then, the distributed algorithm, a main contribution of this work, is elaborated as a practical application. The Huffmies concept emphasizes the following three issues: (a) minimizing delays between the content server and end-users, (b) maximizing the transmission speed, and (c) enhancing ISP gateway-link fault immunity. Based on the presented simulations, we illustrated that the proposed distributed Huffmies approach achieves better results than other concepts, especially when a swarm aggregates more peers and the ISP networks become congested. Since the approach is beneficial for end-users, content providers and also ISPs, we can argue that the usage of Huffmies is fit for all entities involved in the overlay video multicast.

The properties of our approach are based on its similarity to the Huffman source coding providing the optimal t-ary tree. Likewise, we provide the optimal transmission tree for the P2P video streaming basing on the RTT measurements. There are two main challenges related to such an approach.

(1) The node aggravation, that is, the counterpart of the probability in the Huffman algorithm, calculated on the basis of the RTT measurements, might change in time due to the well-known measurement problems (e.g., the results are unstable). We counteract the negative consequences by repeating the measurements and rebuilding the tree. Additionally, the nodes are grouped in the distributed version of Hummies in such a way that the delay measurements are more precise, since they are related to peers located close to each other.

(2) The Huffman coding assumes that each intermediate node should have t children to attain the optimality, but in the case of some peers in the Huffmies tree, the assumed number of the attached children might harm

the performance due to lack of resources. This problem can be dealt with by letting the nodes "cheat" if they have insufficient resources. The cheating consists in informing that some of their t transmission slots are used, while in fact they are not.

Acknowledgment

This paper is an extended version of [21]. The work is supported by the Polish National Science Centre under Grant N N517 555239.

References

[1] S. Tang, Y. Lu, J. M. Hernández, F. Kuipers, and P. van Mieghem, "Topology dynamics in a P2PTV network," in *Proceedings of the IFIP-TC6 International Conference on Networking (NETWORKING '09)*, Aachen, Germany, May 2009.

[2] R. J. Lobb, A. P. Couto da Silva, E. Leonardi, M. Mellia, and M. Meo, "Adaptive overlay topology for mesh-based P2P-TV systems," in *Proceedings of the 19th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '09)*, pp. 31–36, Williamsburg, Va, USA, June 2009.

[3] F. Picconi and L. Massoulié, "ISP-friend or foe? Making P2P live streaming ISP-aware," in *Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS '09)*, Montreal, Canada, June 2009.

[4] W. Liang, J. Bi, R. Wu, Z. Li, and C. Li, "On characterizing PPStream: measurement and analysis of P2P IPTV under large-scale broadcasting," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '09)*, Honlulu, Hawaii, USA, November-December 2009.

[5] E. K. Lua, X. Zhou, J. Crowcroft, and P. van Mieghem, "Scalable multicasting with network-aware geometric overlay," *Computer Communications*, vol. 31, no. 3, pp. 464–488, 2008.

[6] T. Peng, Q. Zheng, W. Lv, S. Jiang, and J. Gao, "Network friendly tree for peer-to-peer streaming," in *Proceedings of the 12th International Conference on Computer Supported Cooperative Work in Design (CSCWD '08)*, pp. 1024–1028, Xi'an, China, April 2008.

[7] R. Besharati, M. Bag-Mohammadi, and M. A. Dezfouli, "A topology-aware application layer multicast protocol," in *Proceedings of the 7th IEEE Consumer Communications and Networking Conference (CCNC '10)*, Las Vegas, Nev, USA, January 2010.

[8] T. Kikkawa, T. Miyata, and K. Yamaoka, "Maximum-bandwidth ALM tree on tree network," in *Proceedings of the 7th IEEE Consumer Communications and Networking Conference (CCNC '10)*, Las Vegas, Nev, USA, January 2010.

[9] I. Rimac, V. Hilt, M. Tomsu, V. Gurbani, and E. Marocco, "A Survey on Research on the Application-Layer Traffic Optimization (ALTO) Problem," IETF RFC 6029, October 2010.

[10] G. Dán, T. Hoßfeld, S. Oechsner et al., "Interaction patterns between P2P content distribution systems and ISPs," *IEEE Communications Magazine*, vol. 42, no. 5, pp. 222–230, 2011.

[11] J. Seedorf and E. W. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement," IETF RFC 5693, October 2009.

[12] O. Abboud, A. Kovačević, K. Graffi, K. Pussep, and R. Steinmetz, "Underlay awareness in P2P systems: techniques and challenges," in *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS '09)*, Rome, Italy, May 2009.

[13] A. Kovačević, O. Heckmann, N. C. Liebau, and R. Steinmetz, "Location awareness-improving distributed multimedia communication," *Proceedings of the IEEE*, vol. 96, no. 1, pp. 131–142, 2008.

[14] X. Tu, H. Jin, X. Liao, and J. Cao, "Nearcast: a locality-aware P2P live streaming approach for distance education," *ACM Transactions on Internet Technology*, vol. 8, no. 2, 2008.

[15] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[16] R. Gallager, "Variations on a theme by Huffman," *IEEE Transactions on Information Theory*, vol. 24, no. 6, pp. 668–674, 1978.

[17] D. E. Knuth, "Dynamic Huffman coding," *Journal of Algorithms*, vol. 6, no. 2, pp. 163–180, 1985.

[18] J. S. Vitter, "Design and analysis of dynamic Huffman codes," *Journal of the ACM*, vol. 34, no. 4, pp. 825–845, 1987.

[19] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of the 15th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '96)*, San Francisco, Calif, USA, March 1996.

[20] J. Tyszer, *Object-Oriented Computer Simulation of Discrete-Event Systems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[21] B. Polaczyk, P. Chołda, and A. Jajszczyk, "Huffman coding inspired peer-to-peer multicasting," in *Proceedings of the 10th International Symposium on Electronics and Telecommunications (ISETC '12)*, Timisoara, Romania, November 2012.