OAZE: a Network-Friendly Distributed Zapping System for Peer-to-Peer IPTV

Yiping Chen, Erwan Le Merrer, Zhe Li, Yaning Liu, Gwendal Simon

OAZE: A network-friendly distributed zapping system for peer-to-peer IPTV

Yiping Chen [a], Erwan Le Merrer [a,c,1], Zhe Li [b], Yaning Liu [b,d,1], Gwendal Simon [b,*]

[a] *Technicolor, 1 avenue de Belle Fontaine, CS 17616, 35576 Cesson-Sévigné, France*
[b] *Institut Telecom, Telecom Bretagne, Technopole Brest Iroise, 29232 Brest, France*
[c] *INRIA Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes, France*
[d] *Beijing University Posts and Telecommunications, 10 Xitucheng Road, Haidian District, Beijing, PR China*

IPTV systems attracting millions of users are now commonly deployed on peer-to-peer (P2P) infrastructures and provide an appealing alternative to multicast-based systems. Typically, a P2P overlay network is associated with each channel, composed of users who receive, watch and redistribute this channel. Yet, channel surfing (aka as zapping) involves switching overlays and may introduce delays, potentially hurting the user experi-ence when compared to multicast-based IPTV. In this paper, we present a distributed sys-tem called OAZE (Overlay Augmentation for Zapping Experience) which speeds up the switching process and reduces the overall cross-domain traffic generated by the IPTV sys-tem. In OAZE, each peer maintains connections to other peers, not only in a given channel, but also in a subset of all channels to which the associated user is likely to zap. More spe-cifically, we focus on the channel assignment problem, *i.e.* determining, in a given P2P overlay, the optimal distribution of the responsibility to maintain contact peers to other channels. We propose an approximate algorithm providing guaranteed performances, and a simpler and more practical one. Our experimental results show that OAZE leads to substantial improvements on the connections between peers, resulting in less switching delay and lower network cost; it then represents an appealing add-on for existing P2P IPTV systems.

1. Introduction

1.1. Context and motivations

P2P systems have recently been considered as appealing alternatives to disseminate TV content over the Internet (aka IPTV). Such approaches have led not only to theoretical proposals (see [1] for an overview of the main theoretical challenges) but also practical applications used by millions of users (e.g., PPlive, sopcast, PPStream). Such system usually rely on creating a P2P overlay network per channel, connecting all users of the channel. However, these P2P IPTV systems usually do not implement an efficient channel zapping feature; the interval between the time a new channel is selected and the time the actual playback starts on the screen, commonly called *start-up delay*, can be prohibitively long. Should measurements highlighting the high frequency of zapping be confirmed [2,3], this could be a serious burden for the success of P2P IPTV systems.

The start-up delay can be broken down in two components: the *bootstrap delay* is the delay between the time a user decides to switch channel and the reception of a first video packet from another peer, and the *player buffering delay* is the delay between the reception of the first video packet and the actual video playback. Measurement studies [4–6] consistently report that the total start-up delays ranges from 10 to 20 s for a popular channel and

* Corresponding author.
E-mail addresses: yiping.chen@technicolor.com (Y. Chen), erwan.lemerrer@technicolor.com (E. Le Merrer), yaning.liu@telecom-bretagne.eu (Y. Liu), gwendal.simon@telecom-bretagne.eu (G. Simon).

[1] Supported by project P2Pimages, of the French Media & Networks cluster.

up to 2 min for less popular ones. In our previous paper [7] (see Supplementary data), we analyzed the bootstrap delay in PPlive, and we identified the major cause of the poor bootstrap performances in current P2P systems: this is due to the absence of reactivity from the first set of peers that a new peer contacts when it joins a channel overlay (less than one fifth of contacted peers actually respond). This set of peers, usually referred as the *peerlist* in the literature, is provided by a *tracker*. In addition to the poor ratio of responses of peers in the peerlist, our measurement also showed that two thirds of the peers that provide content in the first 2 min appear in the peerlist. Therefore, the responsiveness of the initial list of contacts (peerlist) when joining a channel is of the untmost importance for an efficient bootstrap. Precisely, channel surfing is about switching channels frequently.

Recent P2P live streaming systems rely on a gossip-based *topology management system* to connect peers (e.g. [8]) of a given channel. The overlay construction is driven by some guidelines, including a *preference system*, which allows to determine which neighbors a peer should preferentially connect with. The "best" peers that a peer can find in its overlay are called its *matching peers*. Periodically, some potential new neighbors are introduced to every peer by its neighbors. An operation allows to determine if some of these potential neighbors match better than current overlay neighbors. If so, the neighborhood can be improved by simply replacing the less matching overlay neighbors by these new matching ones, then a *handshake* should be completed (including typically an exchange of the map of downloaded packets). A cycle corresponds to the reception of a new list of potential neighbors, and the associated handshakes. The *T-Man* gossip system has shown the benefits one can expect from such a topology management system [8]. From the initial peerlist, a peer iteratively discovers better peers until it is connected to its matching peers. Ideally, the initial peerlist contains only matching peers. With a randomly chosen initial peerlist (a list of supposedly active peers provided by a central server for a given channel), a joining peer has low chance to be connected directly to matching peers. Despite some works in this direction (e.g. [9,10]), a central server can hardly provide peers with matching peers quickly, in a scalable fashion. This calls for a new strategy.

In this paper, we present OAZE for *Overlay Augmentation for Zapping Experience*, which has been introduced in [7]. We focus on the case where the preference system of the mesh-based P2P IPTV system is a function of the network distance between peers for that enable to improve startup delays. The design of *network-friendly P2P systems*, where peers tend to connect preferentially to peers that are located in the same Autonomous System (AS), has recently appeared to be a major topic of research [11,12]. We show that OAZE improves the channel switching process by connecting with high probability a peer joining a new channel to peers that are actually active in the overlay of this channel. Furthermore, since the initial peerlist contains matching peers with higher probability than peerlists provided by central servers, connections between distant peers are avoided and the number of handshakes are reduced.

1.2. Rationale

In IPTV systems, some channels are clearly more popular than others [2,13]. Moreover, channels exhibit content similarities such that a participant is more likely to switch to a channel displaying similar content than the one it is currently enjoying. For example, it has been shown [2] that 76% of switches are done in the same channel genre (e.g. sport, music or news). Let us state that a channel c_2 is *adjacent* to a channel c_1 if, when a user watches c_1, the probability to switch to c_2 is among the largest probabilities over all channels. Several works (e.g. [14]) have leveraged this property of users behavior to reduce the switching delay in multicast-based IPTV (networks controlled by telco operators). The idea is to send data of some adjacent channels along with the current channel data, anticipating a potential channel switching. This technique has been extended in order to maximize the probability that the target channels are within the set of adjacent channels [15,16]. In addition, sophisticated recommendation systems [17,18] based on user profile have recently been designed in order to accurately predict the adjacent channels.

Yet, in P2P IPTV systems, receiving simultaneously several multimedia flows, even degraded, remains expensive (important overhead of applicative multicast over IP, compared to lower layer multicast). However, we believe that this ability to predict accurately the most probable switching channels can be leveraged in P2P IPTV systems as well. This is precisely the idea of OAZE, where every peer maintains connections to peers in adjacent channels to account for a potential channel switching operation. More precisely, peers maintain accurate peerlists for the overlays of adjacent channels. In related works related to multi-channel systems [9,19], concurrent to our work, each peer is also connected to one or more channels that are independent of the channel the peer is viewing. In order to limit the bandwidth overhead, each peer is allowed to determine the fraction of upload capacity dedicated to every other stream. This work is complementary to ours. We choose to not impose peers to download unnecessary stream data. Instead, we focus on the connections between peers and their contact peers in adjacent channels. We aim at showing the benefits one can expect from a relevant mechanism to link peers from distinct channels. Yet, it is possible to implement, on top of OAZE, some prefetching mechanisms like the one presented in [9,19].

1.3. Contributions

We have introduced the main concept of OAZE in a previous paper [7]. In this paper, we give a comprehensive description of the OAZE algorithms augmented with two additional contributions: (1) the distribution of the switching task and (2) and extensive evaluation of the system. In OAZE, peers maintain a set of *overlay neighbors*, with whom they exchange video content related to their channel and *contact peers*, peers acting as contact points to other channel overlays. When a peer x has a contact peer in the channel c, we say that x is a *switcher to* c. Instead of being a switcher to all adjacent channels, which would yield

important maintenance costs, peers in OAZE leverage their overlay neighbors to switch to other channels. When a peer x wants to switch to a channel c, it requests contact peers in c from an overlay neighbor that is a switcher to c. Although OAZE has been designed for channel switching, its applicability goes beyond. More generally, the problem addressed in this paper consists of switching from a highly connected cluster of peers to another highly connected cluster within a giant overlay network. Typically, switching from one chapter to another chapter in a P2P VoD streaming system can be formulated in the same way, and solutions are unsurprisingly related to prefetching of most probable seeking positions [20,21].

In this paper, we first address the problem of distributing the switching task to the peers. The performances of OAZE highly depend on the ability of peers to discover a switcher to a given channel in their neighborhood (contact peers). When a peer joins an overlay, it should decide which adjacent channels it wants to become a switcher to, with regard to its overlay neighbors. We call the problem of distributing the switching responsibility to peers in a given overlay the *channel assignment problem*. We show that determining the optimal solution of a channel assignment problem is a NP-hard problem. From a theoretical side, no exact solution can be computed in a reasonable time, and therefore we provide two heuristics to assign adjacent channels to peers: an approximate algorithm and a practical distributed solution. In the first approach, the assignment computation can be distributed. When the number of adjacent channels is α and only one channel is assigned to every peer, the solution is guaranteed to be no more than $\frac{3}{2}\alpha - \frac{5}{2}$ times the optimal one. However, since all peers in the overlay should simultaneously execute the algorithm, this algorithm can only be used in certain cases, for example when the channel assignment is computed from scratch, or when the set of adjacent channels is re-defined. In the second approach, every peer joining a new channel explores its neighborhood in order to determine which channels it should be a switcher to. We show that this algorithm outperforms a basic random algorithm where every peer picks at random the channels it wants to become a switcher to.

Second, we provide a comprehensive set of simulations for OAZE in the case where the P2P IPTV system implements a gossip-based network-friendly topology management system. We show that OAZE significantly improves the quality of the initial peerlists. In practice, these improved peerlists allow to reduce the time to get the first video packets (bootstrap delay reduction) and to decrease the overall impact of the P2P IPTV system on the network. We focus on this latter point in our simulations. Our experiments are conducted over a realistic simulation environment where the network contains multiple ASes. We measure the reduction of the cross-domain traffic that can be expected from the integration of OAZE in an IPTV system. More specifically, we highlight the fact that the overall network cost (measured here as the number of traversed ASes) can be reduced by 10%.

The rest of this paper is organized as follows. Section 2 presents OAZE. Section 3 formulates and analyzes the channel assignment problem, and further presents our algorithms to create switchers. Section 4 reports on the set of simulations, evaluating OAZE using specific network settings. Section 5 concludes the paper.

2. Distributed switching systems and our approach OAZE

We assume an IPTV system consisting of α channels, each channel being a P2P overlay, which is modeled as a graph where the nodes are the peers and the edges represent the mutual knowledge of both peers. The set of all P2P overlays is also a graph noted $\mathbb{G}$. A peer x is involved in exactly one channel among the α channels simultaneously offered by the IPTV service. In this P2P overlay, x cooperates with some peers, called *overlay neighbors*, with the exclusive goal of exchanging content related to its channel. This small set of neighbors is noted $\Gamma^{over}(x)$. Furthermore, let $d_{over}(x,y)$ denote an overlay distance that presents the hop-distance of the shortest path between a peer x and another peer y in the global graph $\mathbb{G}$ (with $d_{over}(x,y) = \infty$ if the channel of x is different from the channel of y). For any integer $k > 0$, let the *k-neighborhood* of x be $\Gamma^{over}(x)_k = \{y | 0 < d_{over}(x,y) \leqslant k\}$, where y is called *k-neighbor* of x.

OAZE is agnostic of the algorithms that are implemented for the P2P overlay, and it can be implemented over any P2P IPTV. However, OAZE is especially designed to fit with the characteristics of mesh-based P2P systems where a gossip algorithm makes that peers are eventually connected to their matching peers. Our assumption is that, in a given P2P overlay, the overlay neighborhood reflects the matching, in other words, for two integers k_1 and k_2 with $k_1 < k_2$, if y_1 is a k_1-neighbor of x and y_2 is a k_2-neighbor of x, then it means that y_1 matches better with x than y_2.

We also care on the impact of the P2P IPTV system on the underlying network. We call *underlying distance* between x and y the distance in the physical network between these two peers. The underlying distance can refer to the latency between x and y, the number of routing hops on the path or the number of AS hops. In this paper, we consider the underlying distance as the number of ASes traversed in the shortest path between x and y in the Internet. It is denoted as $d_{AS}(x,y)$. That is, $d_{AS}(x,y)$ corresponds to the cross-domain traffic generated by any data transfer between x and y. We choose this definition of underlying distance because the cross-domain traffic represents an increasing concern for network operators.

2.1. Our proposal

A peer y is a *contact peer* of another peer x when y is not in the same overlay, but $x \rightarrow y$ relation exists. We note $\Gamma^{inter}(x)^c$ the set of contact peers of x that are in the channel c. A peer x is requested to maintain fresh lists of contacts (dead nodes or nodes that switched are removed from contact lists), and to try to match with those contacts.

We associate with a peer x a set $\mathcal{C}(x)$ of channels such that a channel c belongs to $\mathcal{C}(x)$ if and only if there exists a peer y such that both y is in c and $y \in \Gamma^{inter}(x)^c$. We say that x is a switcher to c. Because the number of channels

α is expected to be large, a peer cannot be a switcher to all other channels. For scalability issues, we bound by δ the number of channels every peer can be a switcher. We depict the overall P2P overlay switching system in Fig. 1.

An *overlay switch* for a node x is the process leading to a complete change of its neighborhood $\Gamma^{over}(x)$, and to another one reflecting its move to a new chosen channel. Say that a peer x in c_i wants to switch to a channel c_j; we describe now the two ways to switch overlays.

First, the k-neighborhood of peer x contains a peer y, which is a switcher to c_j, thus $c \in \mathcal{C}(y)$ and $y \in \Gamma^{over}(x)_k$. Search could be implemented through basic expanding ring search technique, at k hops, from the requesting peer. A peer only has to ask y for its contact list from c_j, update its neighborhood with these peers, and join the targeted channel. Note that, if both P2P overlays c_i and c_j implement the same matching preference system (say peers choose preferentially their neighbors in their own AS), the contact peers of y are close to y, and y is close to x, so there is good chance that the contact peers of y are close to x too. This will speed up the convergence process.

The second scenario is simply a failure in searching distributively a contact peer for the targeted overlay. It can occur either because no switcher is accessible at k hops, or because the chosen channel is not an adjacent one. In such a case, the traditional centralized approach is used to get the initial peerlist. Note that in IPTV, servers are mandatory, as they are also in charge of pushing the content into overlays, and to serve as initial bootstraps for the IPTV service.

2.2. Main algorithms

Mesh-based P2P overlays use gossip algorithms to ensure that the topology conforms a *preference system*. In the literature related with such gossip-based topology management systems, it has been shown that it is important to exchange a list of peers not only with the overlay neighbors, but also with peers that have been randomly chosen in the P2P overlay. Typically, in the T-man system, only a few cycles are needed to reach a near optimal peer matching, even in presence of churn [8]. The random peers needed by the algorithm are provided by a *peer sampling protocol*. It has also been shown that the gossip paradigm can implement efficient peer sampling protocols, see e.g. Cyclon [22] or the Peer Sampling Service [23].

We distinguish two classes of preference systems. In a global preference system, each peer can be associated with a unique value reflecting a quality metric, and therefore two neighbors can be compared by a simple distance computation. In a local preference system, each peer ranks other peers from its own point of view (e.g. network latency). In OAZE, peers use the same preference system for the management of both their overlay neighbors Γ^{over}, and their contact neighbors Γ^{inter}, except that this procedure is not bidirectional (the switcher tries to match with peers in target overlay, but the reverse case may not be true). With this matching process, once a peer wants to switch, and is able to find a switcher in its k-neighborhood, the contact peers provided by the switcher have a high chance to be close from its future position in the new channel.

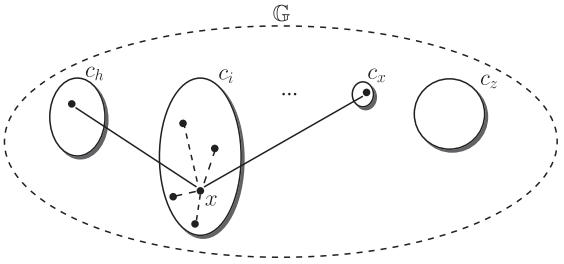


Fig. 1. An example of a resulting system $\mathbb{G}$. Node x has 4 neighbors in its current overlay c_i (dashed lines) and 2 contacts in other overlays, $\delta = 2$ (plain lines). Node x is a switcher to channels c_h and c_x. If one of its direct neighbors in overlay c_i wants to switch to either c_h or c_x, it can leverage x to discover matching and active peers in this overlay.

Algorithm 1. The OAZE algorithm

Initially: upon IPTV join of a node x in channel c_i
1: arbitrarily fixed input: k (depth of switcher search)
2: request server c_i that returns a peerlist $\Gamma^{over}(x)$
Matching for x in current channel:
3: stabilization and convergence toward matched $\Gamma^{over}(x)$
4: execute *channel assignment algorithm* determining $\mathcal{C}(x)$
5: for every channel c_j in $\mathcal{C}(x)$ do
6: request server c_j that returns a peer list $\Gamma^{inter}(x)^{c_j}$
7: stabilization and convergence toward a matched $\Gamma^{inter}(x)$
Upon switch to channel c_j:
8: execute *switcher lookup algorithm*
9: if there exists a switcher y to c_j in $\Gamma^{over}(x)_k$ then
10: $\Gamma^{inter}(y)^{c_j}$ becomes $\Gamma^{over}(x)$
11: else
12: request server c_j that returns a peerlist $\Gamma^{over}(x)$
13: goto line 3

We present in Algorithm 1 a global description of OAZE. This algorithm contains two key algorithms: the *channel assignment algorithm* and the *switcher lookup algorithm*. This latter returns a switcher to a channel if there exists one in the k-neighborhood. Various classic algorithms can be implemented (e.g. flooding with increasing depth). We now focus on the channel assignment algorithm, because it is a critical component of OAZE.

3. Channel assignment

In this Section, we analyze the problem of assigning a set of channels to peers so that every channel is easy to be accessed from every peer. Note that, this problem applies when a small set of resources should be distributed

among a large set of peers. Hence, this analysis presents a scientific interest that goes beyond the performances of OAZE.

We illustrate the problem in Fig. 2, where six nodes (1, 3, 4, 7, 8 and 11) form an overlay network for the channel *a*, two nodes (2 and 9) in channel *b*, two nodes (5 and 10) in channel *c* and node 6 in channel *d*. We represent the linkage in underlay network on the bottom, and the channel assignment for the channel *a* on the top.

This channel assignment has a drawback: the two direct overlay neighbors of node 11 are switchers to the same channel *b*. Therefore, for *k* equal to one, the node 11 is unable to discover a switcher to *c*. We derive from this failed assignment the main objective of a channel assignment algorithm, which is to maximize the number of distinct channels for which there exists a switcher in the *k*-neighborhood of every peer. This is objective 1.

Let us now observe the case of node 1, when *k* is equal to two. If this node wants to switch to channel *b*, two switchers can be used: nodes 4 and 8. The node 4 is the ideal switcher to channel *b* because node 4 is close to node 1 in the underlying network. If switchers are connected to their matching contact peers in channel *b*, node 4 would be able to introduce node 1 to node 2, which is its matching peer in channel *b*. We derive another definition of the assignment objective, which is to minimize the underlying distance to a switcher to every channel. This is objective 2.

3.1. Maximizing the number of channels in k-neighborhood is NP-hard

We first show that the problem defined with objective 1 is NP-hard. In domination theory [24], a set $D \subseteq V$ of vertices in a graph (V, E) is called a *dominating set* if, for every vertex *x* in V, *x* is either an element of *D* or is adjacent to an element of *D*. More generally, a *k-dominating set* extends this adjacency notion at *k* hops, at most, from a given peer *x*. Consider that at most δ resources (here channels to be switcher to) can be allocated to a node. A δ-*configuration* is an allocation of a set of resources such that, for every resource, the vertices associated with this resource form a dominating set. The same extension as in previous definitions can state for a δ-configuration. That is, a (δ, k)-*configuration* is to allocate resources to vertices, such that no more than δ different resources are allocated to any vertex, and each vertex can access a resource associated with another vertex in less than *k* hops. Applied to the switcher creation context, the resources are channels, nodes associated with a given resource are switchers for this channel.

Unfortunately the decision problem associated with the (δ, k)-configuration is NP-complete. All optimization problems that are directly related to this decision problem are NP-hard. The (δ, k)-configuration problem is the problem of assigning channels to peers so that every peer has a switcher for all channels in its *k*-neighborhood. This NP-hardness result makes that maximizing the number of channels α that can be allocated, with a given δ and a given *k* is NP-hard, as well as minimizing the number of switcher peers δ for a given *k* and a given α. Therefore, an optimal solution cannot reasonably be computed in a large-scale system.

3.2. Our formulation of the channel assignment problem

We propose a slightly different formulation of the problem, which combines both objectives 1 and 2, but neglects the parameter *k*. We focus on assigning channels so that we minimize a *distance* from every peer to its closest switchers. Here, the distance, noted *d*, is a generic notion, which can be associated with the overlay distance d_{over} (if the objective is actually to reduce the overlay distance between a peer and its switchers) or with the underlying network distance d_{AS} (if the objective is to build a network-friendly P2P IPTV system), or with any mix of both distances.

The *rainbow distance* for a peer *x* is the sum of the distances from *x* to the closest peers hosting collectively the α adjacent channels. We denote by $d(x)$ the rainbow

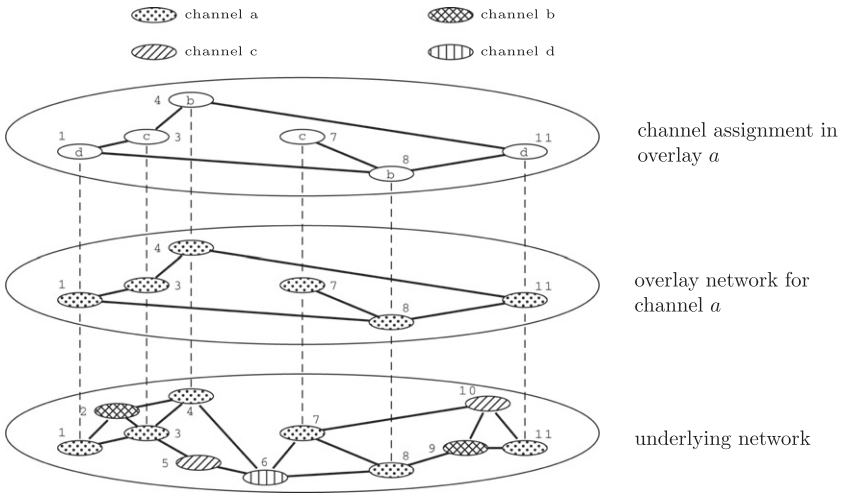


Fig. 2. An example of underlying network and overlay network.

distance of a peer x. The goal is to reduce the sum of all rainbow distances. In other words, we want to minimize the average distance for each peer x to access a switcher for every adjacent channel. Let V be the set of peers in a given channel. Let note by $\mathbb{G}_{adj}$ the α adjacent channels of this channel. The goal is to determine an allocation $\mathcal{C}$ from adjacent channels to peers such that:

$$\sum_{x \in V} \sum_{c \in \mathbb{G}_{adj}} \min \{d(x,y) : x \in V, c \in \mathcal{C}(y)\}.$$

This problem is closely related with a variant of Facility Location Problems, namely the α-Product Uncapacitated Facility Location Problem (α-PUFLP). In this problem, we have a set of facilities and a set of clients. Each client has an access cost to retrieve the service deployed at any facility. In this variant, the service consists of α distinct products. The objective is to open a subset of the facilities and to satisfy all demands of clients so that the total cost is minimized. Few papers have dealt with this NP-complete problem [25–27]. An approximate solution has been proposed, where the solution is at most $\frac{3}{2}\alpha - 1$ times the optimal one [28]. To some extent, our problem can be seen as a sub-problem of α-PUFLP as it is possible to transform any instance of our channel assignment problem into an instance of α-PUFLP. Indeed, every peer can be seen as both a facility and a client with a null access cost between them.

Please note that this problem is for one channel, and the algorithm should be applied to every channel in the P2P IPTV system. Please finally note that, when $\delta > 1$, we regard one peer as δ peers located at the same place. Therefore, in the following subsections, we only consider the case where $\delta = 1$.

3.3. Distributed approximate algorithm

We first present a distributed algorithm that computes a channel assignment with guaranteed performances. This algorithm computes from scratch a full assignment, and it should be executed by all peers. Therefore, in practice, this algorithm does not cope with dynamic systems. In a controlled environment, this algorithm can be executed on a periodic basis in order to refresh the assignment.

We introduce some additional notations. From now on, the nodes that serve a peer x (i.e. the switchers to every channel that are the closest to x) are noted $F(x)$. Besides, the $\alpha - 1$ nearest neighbors of a peer x are noted $N(x)$, while $N[x] = N(x) \cup \{x\}$. Moreover, we denote by $N(x)_2$ the union of the $\alpha - 1$ nearest neighbors of x and their respective $\alpha - 1$ nearest neighbors. Note that, in an ideal scenario, $F(x) = N[x]$ but this solution is sometimes impossible.

We compute a *fractional optimal solution*, an impossible solution, which gives a lower bound of the optimal solution. We cut each channel into α fractional pieces. Then, we form the fractional optimal solution S by assigning to each x in V a fraction $\frac{1}{\alpha}$ of all the channels. Therefore, each peer can have an access to any channel c by retrieving a fraction of c to itself, and the remaining $\frac{c-1}{c}$ fractions from its $\alpha - 1$ nearest nodes in the overlay. The solution S is a fractional optimal solution because the sum of fractions assigned to each node is equal to 1 and all channels can be accessed from itself and some neighbors in the overlay, that is, $F(x) = N[x]$. We denote the rainbow distance of a node x in the solution S as $\bar{d}_x$.

Our distributed algorithm is inspired by Li and Li [28]. Details are given in Algorithm 2. The main idea is that every peer tries to optimally assign channels to their nearest peers and themselves. In order to avoid that two peers assign a channel to the same peer simultaneously, each peer x initially exchanges information with all peers in $N(x)_2$ (lines 1–2). Then, these 2-hop neighbors are sorted in increasing optimal rainbow distance $\bar{d}$ order (line 3). Once x has received messages from all peers in $N(x)_2$, it enters in *waiting mode*.

Algorithm 2. Approximate Algorithm for Channel Assignment (node x)

```
      Initially:
1:    determine N(x)_2, compute the optimal rainbow
      distance d̄_x
2:    broadcast d̄_x to every peer in N(x)_2
3:    create a list L(x) containing all peers in N(x)_2
      sorted in increasing rainbow distance order
4:    enter waiting mode
      In waiting mode:
5:    upon reception of a release message from a node y
6:    discard y from L(x)
7:    if x is the first peer in L(x) then
8:      let C_toAssign = G_α
9:      let N_toAssign be all peers in N[x] without any
        assigned channel
10:     for all nodes y in N[x] \ N_toAssign do
11:       remove C(y) from C_toAssign
12:     if card(N_toAssign) = card(C_toAssign) then
13:       assign channels in C_toAssign to peers in
          N_toAssign such that no pair of peers are
          assigned the same channel
14:     broadcast release message to every peer in
        N(x)_2
15:     enter in final mode.
      In final mode:
16:   upon reception of a release message from a node y
17:   discard y from L(x)
18:   if L(x) is empty and no channel is assigned then
19:     assign to x the channel whose closest switcher
        is the farthest among all channels
```

The peers with minimal optimal rainbow distance among their 2-hops nearest neighbors execute the algorithm first (lines 5–7). When a peer cannot produce a local optimal assignment, it leaves itself unassigned. Such a configuration occurs when at least two peers among the $\alpha - 1$ nearest neighbors have already been assigned the same channel. Otherwise a peer assigns channels to all its unassigned nearest neighbors and itself (lines 8–13). Finally, a peer sends a *release* message and enters in *final mode*.

In the final mode, some peers may be still unassigned. It means that neither they, nor their nearest neighbors were able to optimally assign channels. In this case, a peer assigns itself the channel that is the farthest (lines 18–19).

Theorem 1. *For any $\alpha \geqslant 3$, Algorithm 2 gives an integer solution no more than $\frac{3}{2}\alpha - \frac{5}{2}$ times of the fractional optimal solution for the Channel Assignment Problem.*

Proof. For every optimized peer x, we know that $d_x = \bar{d}_x$. When x cannot be optimized after the *waiting mode*, we distinguish two cases.

In the first case, the channel on x has been assigned by an optimized peer $x' \in N(x)$, but this channel conflicts with the channel hold by another peer in $N(x)$. Since x' was in *waiting mode* before x, we know that $\bar{d}_{x'} < \bar{d}_x$. As x' has been optimized, all channels are assigned among its nearest neighbors. Assume that the channel on x is c_1. For the node x, a way to reach every other channel $c_2, \ldots, c_\alpha$ is to use the neighbors of x'. Let note y_i the node in $N[x]$ that has been assigned channel c_i, for $2 \leqslant i \leqslant \alpha$. We have:

$$\begin{aligned}\sum_{y \in F(x)} d(x,y) &\leqslant \sum_{i=2}^{\alpha} d(x,y_i') \leqslant \sum_{i=2}^{\alpha} (d(x,x') + d(x',y_i')) \\ &\leqslant (\alpha-2)d(x',x) + d(x',x) + \sum_{i=2}^{\alpha} d(x',y_i) \\ &\leqslant (\alpha-2)d(x',x) + \bar{d}_{x'}.\end{aligned}$$

Since the distance $d(x',x)$ is one of the $(\alpha-1)$ distances that are part of the rainbow distance of x', we have $d(x,x') \leqslant \bar{d}_{x'}$. We also know that $\bar{d}_{x'} < \bar{d}_x$, therefore, we obtain:

$$\sum_{y \in F(x)} d(x,y) \leqslant (\alpha-1)\bar{d}_x. \tag{1}$$

In the second case, two nodes in $N(x)$, say y and y', have been assigned the same channel. Since these nodes hold a channel, there exists an optimized node x' at 2 hops from x, with $\bar{d}_{x'} < \bar{d}_x$. Assume without loss of generality that y and y' have the channel c_1 and that $d(x,y) \leqslant d(x,y')$. After the *final mode*, the peer x picks a channel, say c_α, that is different from c_1 because c_1 is not the farthest channel to access for x. Let y be in $N(x) \cap N(x')$.

When all peers are assigned a channel, since x and y are assigned a different channel, node y is included in $F(x)$. We use the same idea as in the previous case to express the rainbow distance of the node x as the sum of the distance from x to x' plus the rainbow distance of x'

$$\begin{aligned}\sum_{y \in F(x)} d(x,y) &\leqslant d(x,y) + (\alpha-2)(d(x,y) + d(y,x')) + \sum_{c=2}^{\alpha-1} d(x',y_c') \\ &\leqslant (\alpha-1)d(x,y) + (\alpha-3)d(y,x') + \bar{d}_{x'}.\end{aligned}$$

Since $d(y,x')$ is part of the rainbow distance of x', we have $d(y,x') \leqslant \bar{d}_{x'}$, and therefore $(\alpha-3)d(y,x') + \bar{d}_{x'}$ is lesser than or equal to $(\alpha-2)\bar{d}_{x'}$. Moreover, since $d(x,y) < d(x,y')$ and both distances are parts of the optimal rainbow distance $\bar{d}_x$, we have:

$$(\alpha-1)d(x,y) \leqslant \frac{\alpha-1}{2}(d(x,y) + d(x,y')) \leqslant \frac{\alpha-1}{2}\bar{d}_x$$

Hence we obtain:

$$\sum_{y \in F(x)} d(x,y) \leqslant \left(\frac{3}{2}\alpha - \frac{5}{2}\right)\bar{d}_x. \tag{2}$$

As $\alpha - 1 \leqslant \left(\frac{3}{2}\alpha - \frac{5}{2}\right)$ for any α greater than 3, then Algorithm 2 gives an integer solution no more than $\frac{3}{2}\alpha - \frac{5}{2}$ times the fractional optimal solution. □

3.4. Practical algorithms

The advantages of Algorithm 2 include its guaranteed performances. However, all peers in the overlay should execute the algorithm simultaneously. We propose two simpler algorithms, which can be executed by any peer at any time. However, their performances are not guaranteed.

In the first algorithm, every peer selects the channel whose switcher is the farthest from itself, and it assigns this channel to itself. We call this algorithm *delta-k* because a peer explores its nearest neighbors by increasing flooding.

In the second algorithm, which is the simplest one, every peer assigns to itself a channel that is randomly picked among the adjacent channels.

3.5. Comparative simulations

This first set of simulations aims to evaluate the distinct algorithms and to determine the best choice for the remaining of the paper. These simulations are in three parts, which correspond to three distinct scenarios. We had 1000 peers, which were randomly located in 200 different ASes. We used a real Internet map, and the inter-AS distance was measured by a technique that is detailed in Section 4. The number of adjacent channels was fixed at 15. All results are given in Fig. 3.

In the first scenario, we did not take into account the overlay. The distance d only depended on the AS distance between peers. This simulation corresponds to the scenarios where the number of peers in every channel is not large, or when the number of overlay neighbors is larger than the number of channels. The results for this scenario are figured as *x-Algo* where *Algo* can be *Rand* for the random algorithm, *delta* for the delta-*k* algorithm, and *approx* for the approximate algorithm.

We compared the average distance to the closest switcher to every channel for the three algorithms. Both *approximate* and *delta-k* algorithms outperform the *random* algorithm (more than 15% of gains). Moreover, we show that *delta-k* performs approximately as well as the *approximate* algorithm, despite it can be executed asynchronously.

In the second and the third scenarios, we took into account an overlay. The average overlay degree was fixed at 5. For both scenarios, the *approximate* algorithm was built as follows: every peer selected its $\alpha - 1$ nearest neighbors in terms of AS distance among the peers in its 3-neighborhood. For the *delta-k* algorithm, the peer explored its neighborhood until it found switchers to all channels, then it assigned the channel that was the farthest in term of AS distance.

In the second scenario, the overlay was built at random. It is the worst case where the P2P IPTV system does not incorporate any network-friendly mechanism. The results

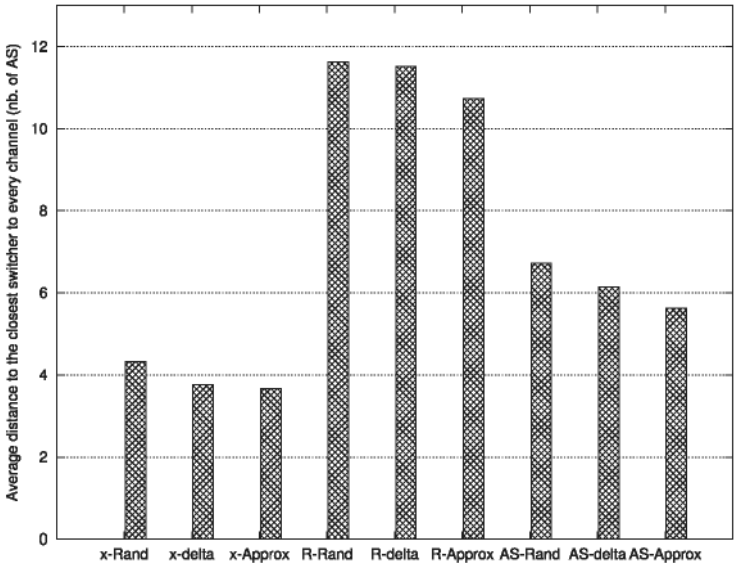


Fig. 3. Comparison ot three channel assignments. The prefix *x*– stands when the distance function does not take into account the overlay, *R* for a random overlay, AS for a 5-nearest neighbor overlay.

are indicated with *R* (for random overlay) as a prefix. First, the average distance to switchers is more than three times larger than when no overlay constrains peer mutual knowledge. Then we can observe the differences among the algorithms. The *approximate* algorithm outperforms by more than 10% both *random* and *delta-k* algorithms, which have almost the same performances. The main indication of this set of simulations is that no channel assignment algorithm provides any significant gain in term of network-friendliness when the P2P IPTV system does not incorporate any network-friendly process.

In the third scenario, the overlay was built by a perfect network-friendly process so that the overlay was a 5-nearest neighbor graph in the AS distance. Despite the low average degree, this scenario corresponds to a configuration where we can expect good results from the OAZE system. Here, the results indicate that the *approximate* algorithm is approximately 10% better than the *delta-k* algorithm, which is also approximately 10% better than a *random* choice.

3.6. Discussion

The results we obtained from these simulations (based on representative "toy-networks") do not convince us to implement the approximate algorithm in our main simulations. Approximately 15% of expected gains do not appear to be important enough to justify the implementation of an algorithm that requires a global synchronisation from all peers. Therefore, in the following, we will use this *delta-k* algorithm in our OAZE proposal.

4. Simulations using an AS Map

We evaluated our OAZE proposal using the PeerSim simulator [29]. The multi-channel system we implemented as an input conformed recent application measurements conducted in [2]. The aim of this section is to put light on the benefits, considering the cross-domain traffic, that could be awaited from OAZE if implemented in a mesh-based P2P streaming system having a gossip-based topology management.

4.1. Local preference system based on AS distance

We first explain our implementation of the local preference system based on the AS distance between peers. This preference system was a pre-calculated $N \times N$ matrix. Each line of this matrix defines the preference rank of all other peers with respect to one peer. A small value at point (x, y) stands for a high attractivity for peer x to be matched with y.

We adopted the AS level hop count as the rank value. These hop counts were calculated using the CAIDA dataset [30], a widely adopted trace representing a realistic view of Internet at the AS level. This dataset contains 28，421 ASes and their type of relationship (*peering* or *transit*). Two ASes *AS*1 and *AS*2 have a transit relationship if *AS*1 transits the traffic from *AS*2 and charges *AS*2. The transit, which is a provider–customer relationship, is commonly observed between big and small ASes, as small ASes need big ones to be connected to the Internet. In the second type of relationship, peering, a settlement is negotiated between *AS*1 and *AS*2 so that they can exchange traffic for free. This generally happens between two ASes of same size and importance.

We followed the algorithm given in [31] to compute the relationship-based routing distance on top of this dataset. When an AS receives a routing request, it takes a local decision for next hop (often called a *greedy* decision) based on its relationship with the neighboring ASes. The request is forwarded in decreasing preference order to: (i) customer

AS, (ii) peering AS, (iii) provider AS. Note that a route is not necessarily used both ways, so the matrix is asymetric. Fig. 4 shows the hop-distribution between ASes in the distance matrix. We observe that the distribution of distances between ASes is not uniform. Most of the ASes are located from 6 to 11 hops from each other, while the distance is significantly higher for some other fractions of ASes.

This knowledge on AS relationship allows us to obtain a finer grained estimation of the impact of our approach on the network cost. Peers now construct and update their sets of overlay neighbors Γ^{over} and contact peers Γ^{inter} by preferring neighbors with the smallest AS distances available. The goal is to reduce the whole cross-domain traffic, and more specifically the transit traffic because it implies financial transactions.

4.2. Simulation configuration

We considered a 150 channel system. According to the measurements of [2], channels belonged to one of 13 genres (e.g. cine, sports, kids, docu). Channel popularity in each genre followed a Zipf distribution, that decays fast for non-popular channels. The probability of a peer to remain in the same genre during a switch was set to 76% [2,3]. Otherwise, a peer picked uniformly at random first the genre it switched to (probabilities are exposed in [2]), and then the channel in that genre. Fig. 5 shows the number of channel switching for each pair (initial channel, destination channel). We observe different peaks on the figure. Since the channels are grouped in genre by their identifier (from 0 to 149), switching to a channel of the same genre means also switching to a channel with an adjacent identifier. Therefore, most peaks are located close to the diagonal of *xy*-plane.

Each simulation ran for 5000 cycles. A PeerSim cycle execution corresponded approximately to a dozen of seconds in a real system. According to the time peers watched channels, peers get assigned a role: *surfer* (stays at the same channel from 5 to 20 cycles), *viewer* (20–200), *leaver* (200–500). Joining and departing peers created the dynamism (or *churn*) in the system. After each channel switch, a peer picked a role with probability 0.6 for surfer, 0.35 for viewer and 0.05 for leaver.

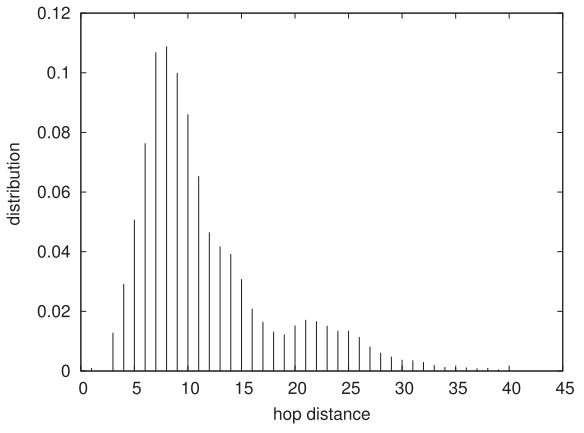


Fig. 4. Distribution of AS-level hop distance.

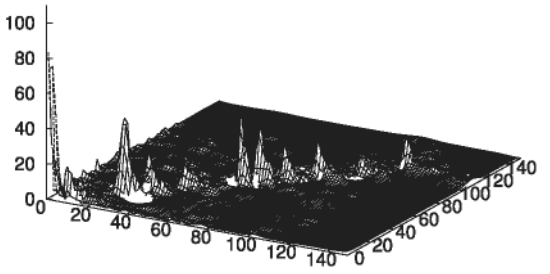


Fig. 5. *x*-axis: Identifier of initial channel, *y*-axis: identifier of destination channel, *z*-axis: number of zapping.

Other simulation parameters were set as follows: 20,000 peers in the system were distributed in 1000 distinct ASes (so, an average of 20 peers per AS). These 1000 ASes were randomly selected among all 28,421 ASes in the dataset. Every peer x had a channel-neighborhood $\Gamma^{over}(x)$ of size β. A peer was a switcher for simply one channel ($\delta = 1$), and was connected to 6 peers in this channel for Γ^{inter}. We added a parameter, μ, that controlled the percentage of peers ($1/\mu$) that were performing the gossip protocols (matching and sampling) at each cycle.

We explored parameter space of μ and β to generate two scenarios: an overlay that quickly converged toward a stable state, where neighboring peers matched (here, $\mu = 4, \beta = 60$), and a slow-converging overlay (with $\mu = 8, \beta = 20$). This is typically of interest for network operators willing to keep some control on the traffic cost, implied by more or less "aggressive" P2P applications.

4.3. Uses of the distributed zapping

At first, we counted the number of times the zapping used our distributed system in comparison to the number of times the central server were used. In Fig. 6, the *x*-axis represents various search depths k allowed to find a switcher. In this simulation, the number of overlay neighbors β, was equal to 20. The ratio of distributed zapping cannot be above 76% for the reason that the 10 adjacent channels represented only 76% of the switch in our simulation, so some zaps must be treated by the central server.

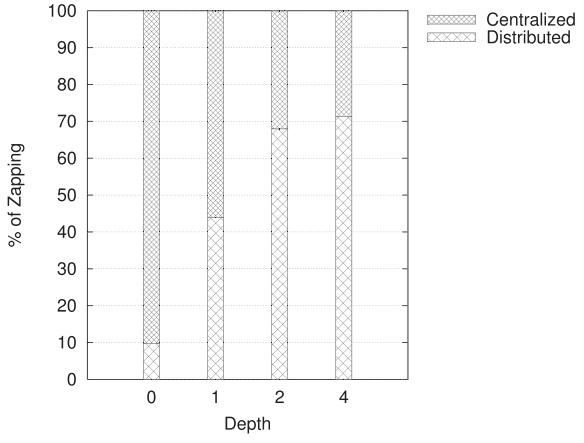


Fig. 6. Number of centralized versus distributed switches.

When $k = 0$, the search was limited to the peer itself: distributed zapping was possible only if the peer wanted to switch to the channel corresponding to its own Γ^{inter}. In this specific case, the number of zaps that require a request to the server is much higher than the number of zaps using OAZE. With the increase of search depth, distributed zapping becomes dominant. The deeper is the switcher lookup, the more channel switches use OAZE.

We also observe that the number of distributed zapping remains almost unchanged between $k = 2$ and $k = 4$. This result highlights that it is possible to find a switcher for most channels at less than three hops. At $k = 4$, OAZE obtains this best achievable result in this specific configuration. However, using a smaller value of k (especially $k = 2$ here) is enough to satisfy a large majority of zaps. Therefore, it is possible to use less costly 2-hops search, or to increase the number of adjacent channels if one can support a 4-hops search.

4.4. Impact of switcher distance on the quality of contact peers

We now evaluate the quality of the peerset that is provided by a switcher depending on the distance between the peer and the switcher. We note $Q_i(x)$ the set of peers that are located at a AS distance exactly i from a peer x, that is, $Q_i(x) = \{y, d_{AS}(x, y) = i\}$. We computed the average distance between x and the peers that are the contact peers of all peers y in $Q_i(x)$. This measurement aims at determining the average distance of the contact peers that are provided by a switcher located at i AS-hops from x. Results are shown in Fig. 7. The x-axis is the AS distance to the switcher, while the y-axis is the average number of AS hops on the route from x to the closest contact peers of switcher at this distance.

As expected, the farther is the switcher, the farther is the contact peer. But some other observations give insights on the characteristics of the inter-AS routing. The transit distance increases surprisingly only until the distance to the switcher is eight, then the transit distance decreases. In other words, the financial gain on the overall traffic is only interesting if the switcher is close to the switching peer.

We also remark that the overall cost quickly increases. For example, a switcher at a distance five gives a contact peers with which the traffic is two times more costly than if the switcher was the peer itself. This result also suggests that the contact node should be selected preferentially in nearby ASes (typically in a AS at less than five hops of the peer).

We have not implemented any mechanism aiming at limiting the contact search process when the distance from the peer is more than a given threshold, but this result highlights that such a limitation may be an interesting future work to explore.

4.5. Distributed zapping efficiency

We now compare the efficiency of distributed and centralized zapping by observing peers' neighborhood Γ^{over} right after each channel switching. We aim at measuring optimality in terms of network distance, this value is the ratio of the sum of distances between a peer and its current neighbors at time t, noted Γ^{over}_t, over the optimal neighborhood Γ^{over}_{opt}, which is the set of nearest peers in the overlay. This neighborhood distance ratio (NDR) of a node x is thus computed by:

$$NDR(x) = \frac{\sum_{y \in \Gamma^{over}_t(x)} d_{AS}(x, y)}{\sum_{y \in \Gamma^{over}_{opt}(x)} d_{AS}(x, y)}.$$

Fig. 8 plots the NDR as a function of time after switch, for two types of settings. We averaged all switches that occur during the whole simulation: $t = 1$ represents the first cycle after switch. We remark that NDR has a high initial value, because of the imperfect neighborhood initially obtained. For centralized zapping, since the cache overlay Γ^{over} is randomly constructed after the zapping (given by the server), The NDR is high. On the contrary, using distributed zapping, the initial Γ^{over} is built using the Γ^{over} of the switcher in Γ^{inter}. These initial contact peers are closer to the switching peer.

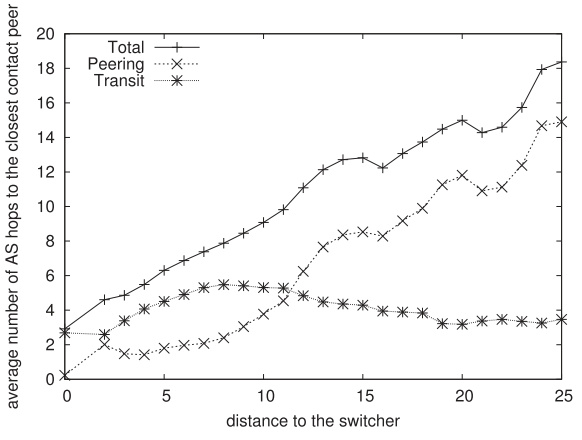


Fig. 7. Average number of AS hop between a peer x and the closest contact peer provided by a switcher y vs. the AS distance between x and that switcher y.

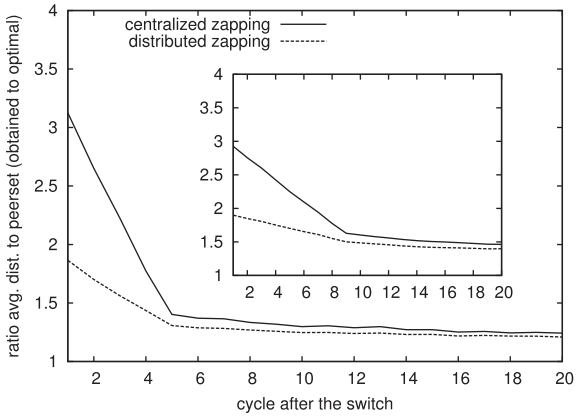
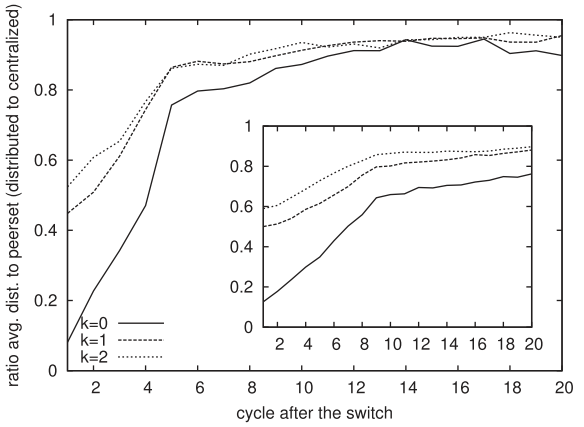


Fig. 8. Comparison of NDR obtained from centralized and distributed switch with two convergence times (main figure: fast converging overlay with $\mu = 4, \beta = 60$; inlay figure: slow converging overlay with $\mu = 8, \beta = 20$).

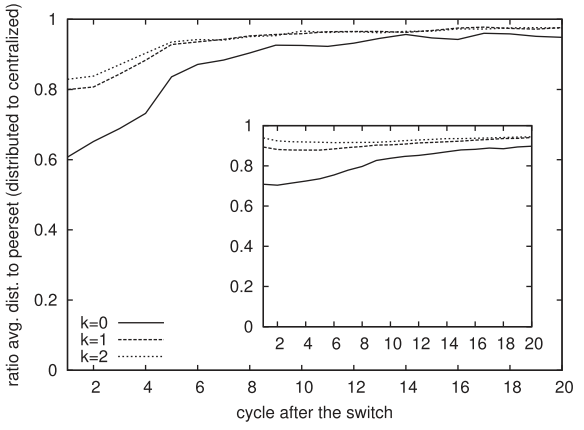
We now want to measure the gain provided by the distributed zapping in comparison to a classic centralized one. For the Fig. 9, we computed the ratio of the average distance to contact peers obtained from a distributed zap to the average distance to contact peers in a centralized zap:

$$\frac{\sum_{y \in \Gamma^{over}_{Distributed}(x)} d_{AS}(x,y)}{\sum_{y \in \Gamma^{over}_{Centralized}(x)} d_{AS}(x,y)}$$

Moreover, we used different values of the switcher lookup depth k, from $k = 0$ to $k = 2$. Thus, we are able to estimate the expected cost saving by using distributed zapping. This figure shows that this ratio is strictly less than one; distributed zapping reduces the peering traffic by 40–90%, and transit traffic by 20–40%, at the time right after the zapping. Then both distributed and centralized methods converge to stable settings because the preference system allows to refresh the neighborhood, so the improvement becomes less significant. Note however that recent measurements have highlighted the high number of peers who switch very frequently between channels. Thus, a watcher is likely to zap before the convergence process ends.



(a) Peering Cost



(b) Transit Cost

Fig. 9. Ratio of the average distance to contact peers in distributed zapping to the average distance in centralized zapping with different k values (main figure: fast converging overlay with $\mu = 4, \beta = 60$; inlay figure: slow converging overlay with $\mu = 8, \beta = 20$).

4.6. Global cost of the P2P-TV system

Finally, we evaluated both distributed and centralized zapping costs from a global point of view. First, we measured the number of handshakes for all peers during the entire simulation. A handshake is a change in neighborhood Γ^{over}, so that the peer should establish a connection to a new neighbor. As we pointed out earlier in the paper, handshakes are costly in terms of time, so they may affect the playback delay of the channel content. We observe in Table 1 that distributed zapping leads to a reduction of the number of handshakes. Since contact peers provided by OAZE are more likely to be matching peers (or peers that are close to matching peers), a switching peers need less handshakes to reach a stable neighborhood.

Secondly, we computed the overall network cost of the system, which is the total network traffic during the whole simulation (still in Table 1). The overall cost is defined as

$$\sum_{\forall x \in V} \sum_{t \in [0,T]} \sum_{y \in \Gamma^{over}_t(x)} d_{AS}(x,y).$$

We can remark that distributed channel switching achieves less peering and transit costs. The system using distributed zapping has reduced the total cost by more than 10% in a slow-converging case.

4.7. Summary

Here are our conclusions from these simulations results:

- Distributed channel switching provides better initial peerset for a switching peer. The average distance to the contact peers is at worst less than two times the optimal although it is more than three for centralized switching.
- Distributed switching significantly reduces both peering and transit costs. This has a practical interest for a network operator managing a P2P IPTV system to reduce the network costs.
- This reduction on network traffic depends on the convergence time of the matching between peers at the overlay scale; higher cost saving can be achieved when a less aggressive peer matching (a longer one) is achieved by the application.

Table 1
Total cost of the P2P TV system.

	Centralized	Distributed	Difference
Fast convergence ($\mu = 4, \beta = 60$)			
Handshakes	1,767,370	1,579,723	−10.62%
Peering cost	15,830,496	12,767,315	−19.35%
Transit cost	64,816,544	63,055,003	−2.72%
Total cost	80,647,040	75,822,318	−5.98%
Slow convergence ($\mu = 8, \beta = 20$)			
Handshakes	310,890	260 914	−16.08%
Peering cost	6,002,171	4,450,306	−25.86%
Transit cost	18,005,011	17,136,952	−4.82%
Total cost	24,007,182	21,587,258	−10.08%

We have demonstrated here the advantage of using a distributed channel switching system like OAZE: a better initial peer set provides less network traffic and decreases the overall cost. Better results can be obtained. For example peers can select several channels ($\delta > 1$), so that the probability to find a nearby switcher becomes higher. Choosing switchers only if they are close to the switching peer appears to be another promising way to improve overall performances.

5. Conclusion

In this paper, we addressed the problem of efficiently switching from one channel to another in P2P IPTV systems, in a distributed fashion. It has been show in literature that initial steps executed by a switching mechanism are crucial for the watcher's experience (reduced delays). To the best of our knowledge, no previous studies have dealt with scalable channel switching, which is a crucial issue for the next generation of multimedia delivery mechanisms over Internet. This approach shows the interest of leveraging peers' belonging to an overlay, in order to improve forthcoming switches. The algorithm proposed in this paper represents a first step toward the design of distributed and efficient switching mechanisms.

References

[1] A. Sentinelli, G. Marfia, M. Gerla, S. Tewari, L. Kleinrock, Will IPTV ride the peer-to-peer stream?, IEEE Communications Magazine 45 (6) (2007) 86.

[2] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, X. Amatrianin, Watching television over an ip network, in: Proceedings of the Usenix/ACM SIGCOMM Internet Measurement Conference (IMC), 2008.

[3] M. Cha, P. Rodriguez, S. Moon, J. Crowcroft, On next-generation telco-managed p2p tv architectures, in: Proceedings of the of International Workshop on Peer-To-Peer Systems (IPTPS), 2008.

[4] X. Hei, C. Liang, J. Liang, Y. Liu, K.W. Ross, A measurement study of a large-scale p2p iptv system, IEEE Transactions on Multimedia 9(8) (2007) 1672–1687.

[5] S. Xie, B. Li, G. Keung, X. Zhang, Coolstreaming: design, theory, and practice, IEEE Transactions on Multimedia 9 (8) (2007) 1661–1671.

[6] B. Fallica, Y. Lu, F. Kuipers, R. Kooij, P.V. Mieghem, On the quality of experience of sopcast, in: Proceedings of International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST), 2008.

[7] A.-M. Kermarrec, E.L. Merrer, Y. Liu, G. Simon, Surfing peer-to-peer iptv: distributed channel switching, in: Proceedings of Euro-Par, 2009.

[8] M. Jelasity, A. Montresor, O. Babaoglu, T-man: Gossip-based fast overlay topology construction, Computer Network 53 (2009) 2321–2339.

[9] C. Wu, B. Li, S. Zhao, Multi-channel live p2p streaming: refocusing on servers, in: Proceedings of the IEEE Infocom Conference, 2008.

[10] A. Boudani, Y. Chen, G. Simon, A quicker way to discover nearby peers, in: Proceedings of ACM CoNEXT Conference, 2007.

[11] H. Xie, Y.R. Yang, A. Krishnamurthy, Y. Liu, A. Silberschatz, P4P: Provider portal for applications, in: Proceedings of the ACM SIGCOMM Conference, 2008.

[12] D. Choffnes, F.E. Bustamante, Taming the torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems, in: Proceedings of ACM SIGCOMM Conference, 2008.

[13] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, J. Xu, Modeling Channel Popularity Dynamics in a Large IPTV System, in: Proceedings of ACM Sigmetrics Conference, 2009.

[14] C. Cho, I. Han, Y. Jun, H. Lee, Improvement of channel zapping time in IPTV services using the adjacent groups join-leave method, in: Proceedings of International Conference on Advanced Communication Technology (ICACT), 2004.

[15] J. Lee, G. Lee, S. Seok, B. Chung, Advanced scheme to reduce IPTV channel zapping time, in: Proceedings of Asia-Pacific Network Operations and Management Symposium (APNOMS), 2007.

[16] D. Boong Lee, H. Joo, H. Song, An effective channel control algorithm for integrated IPTV services over DOCSIS CATV networks, IEEE Transactions on Broadcasting 53 (2007) 789–796.

[17] L. Ardissono, C. Gena, P. Torasso, F. Bellifemine, A. Chiarotto, A. Difino, B. Negro, Personalized recommendation of TV programs, in: Proceedings of Congress on Advances in Artificial Intelligence (AI*AI), 2003.

[18] L. Aroyo, Y. Raimond, D. Brickley, G. Schreiber, L. Miller, The NoTube Beancounter: aggregating user data for television programme recommendation, in: Proceedings of Workshop on Social Data on the Web (SDOW), 2009.

[19] D. Wu, Y. Liu, K.W. Ross, Queuing network models for multi-channel p2p live streaming systems, in: Proceedings of IEEE Infocom Conference, 2009.

[20] C. Zheng, G. Shen, S. Li, Distributed prefetching scheme for random seek support in peer-to-peer streaming applications, in: Proceedings of the ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming, 2005.

[21] Y. He, G. Shen, Y. Xiong, L. Guan, Optimal prefetching scheme in p2p vod applications with guided seeks, IEEE Transactions on Multimedia 11 (1) (2009) 138–151.

[22] S. Voulgaris, D. Gavidia, M. van Steen, Cyclon: Inexpensive membership management for unstructured p2p overlays, Journal of Network and Systems Management 13 (2) (2005) 197–217.

[23] M. Jelasity, A. Kermarrec, M. van Steen, The peer sampling service: experimental evaluation of unstructured gossip-based implementations, in: Proceedings of ACM/IFIP/USENIX International Conference on Middleware, 2004.

[24] T.W. Haynes, S. Hedetniemi, P. Slater, Fundamentals of Domination Graphs, CRC Press, 1998.

[25] J.G. Klincewicz, H. Luss, E. Rosenberg, Optimal and heuristic algorithms for multiproduct uncapacitated facility location, European Journal of Operational Research 26 (1986) 251–258.

[26] J.G. Klincewicz, H. Luss, A dual based algorithm for multiproduct uncapacitated facility location, Transportation Science 21 (1987) 198–206.

[27] H.C. Huang, R. Li, A *k*-product uncapacitated facility location problem, European Journal of Operational Research 185 (2008) 552–562.

[28] B. Li, R. Li, Approximation algorithm for *k*-puflpn, Engineering and Applications 44 (2008) 97–99 (in Chinese).

[29] A. Montresor, M. Jelasity, PeerSim: A scalable P2P simulator, in: Proceedings of International Conference on Peer-to-Peer (P2P), 2009.

[30] The CAIDA AS Relationships Dataset, as-rel.20080107. <http://www.caida.org/data/active/as-relationships/>.

[31] L. Gao, J. Rexford, Stable internet routing without global coordination, IEEE/ACM Transactions on Networking 9 (6) (2001) 681–692.