Cache-to-Cache: Could ISPs Cooperate to Decrease Peer-to-peer Content Distribution Costs?

György Dán
ACCESS Linnaeus Center, School of Electrical Engineering
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: gyuri@ee.kth.se

***Abstract*—We consider whether cooperative caching may reduce the transit traffic costs of Internet service providers (ISPs) due to peer-to-peer (P2P) content distribution systems. We formulate two game theoretic models for cooperative caching, one in which ISPs follow their selfish interests, and one in which they act altruistically. We show the existence of pure strategy Nash equilibria for both games, and evaluate the gains of cooperation on various network topologies, among them the AS level map of Northern Europe, using measured traces of P2P content popularity. We find that cooperation can lead to significant improvements of the cache efficiency with little communication overhead even if ISPs follow their selfish interests.**

I. Introduction

A large share of the Internet's traffic is generated by peer-to-peer (P2P) content distribution systems: an estimated 50 to 80 percent of the total traffic depending on geographic location [16]. For end users, these systems provide quick access to a large variety of content. For content providers, P2P systems provide a means to deliver data to a large population of users without big investments in server capacity and network capacity. The costs of the data delivery are shared among the consumers - the end nodes - and their Internet service providers (ISPs).

The application layer protocols of most P2P systems were not designed to be network aware. Improved network efficiency and the business interests of ISPs are however both strong drivers towards a cross-layer approach in peer-to-peer protocol design: solutions that would decrease operator costs by decreasing the inter-ISP traffic without deteriorating the systems' performance [9].

Proximity aware peer-selection algorithms have been proposed to prioritize nearby peers when up and downloading data [3], [6]. They have been shown to lead to transmission paths with lower round trip times and to reduce cross-ISP traffic, especially for popular contents for which there is a substantial number of peers to choose from. Proximity awareness without ISP support relies on reverse engineering the network topology, e.g., using CDN name resolution [6].

ISP provided application interfaces have been proposed to help proximity aware peer-selection [1], [34]. The application interfaces provide information about the network topology and the network state to the P2P applications, so that the applications can choose more efficient communication patterns than those based on reverse engineered topology information. The proposed systems were shown to increase network efficiency and to decrease ISP costs while not affecting significantly the applications' performance [1], [34].

Proximity awareness can decrease the traffic costs of popular contents, but it cannot decrease the amount of transit traffic if peers cannot be found within the same ISP or a neighboring ISP. P2P caches can decrease the transit traffic costs, and hence, they are complementary to proximity aware neighbor selection [3], [6], [34]. Caches decrease the ISPs' traffic costs by storing local copies of contents, so that data need not to be downloaded from far away peers. P2P caches are available from several vendors, like PeerCache [27], CacheLogic [5] or Oversi [26], and were deployed by many ISPs in recent years. Trace driven simulations [13], [32] and measurements [21] have shown that P2P traffic can be cached efficiently using simple cache eviction policies. Nevertheless, the cache capacity required to achieve high hit rates is considerable, in the order of tens or hundreds of terabytes, because of the heavy tail of the content popularity distribution in P2P systems [13]. Furthermore, the maintenance of P2P caches incurs costs, and hence ISPs are interested in making efficient use of these resources.

Given a number of P2P caches deployed by ISPs, and each ISP following its selfish interest to minimize its transit traffic, we are interested in whether cooperation between the installed caches could lead to benefits for the individual ISPs in terms of decreased transit traffic. The cooperation that we consider consists of *collaborating P2P caches* deployed by peering ISPs: the caches of the ISPs cooperate to serve each others' subscribers and may hence decrease the amount of IP transit traffic. Given the possibly large number of ISPs and caches, the self-interests of ISPs, and the complex AS level peering topology of the Internet, it is not obvious whether cooperation would lead to a reasonably stable allocation of cache resources. It is not clear either how much the ISPs could benefit from cooperation, and how efficient the cooperation between selfish ISPs would be compared to other solutions.

We follow a game-theoretic approach to answer these questions. We model the network of cooperating caches as an n-person non-cooperative game. We consider two models for the caching policies of the ISPs: in the first model the ISPs follow a pure selfish strategy; in the second model ISPs are altruistic, and also consider the interests of neighboring ISPs. Using results from game theory we show that in a system of cooperative caches, both selfish and altruistic, there is always an equilibrium state, a pure strategy Nash equilibrium in game

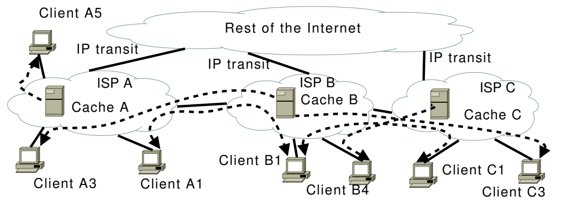

Fig. 1. Cooperative caching and proximity awareness. Three ISPs (A-C), seven clients and five contents (1-5) (Client XY is in ISP X and downloads content Y). P2P clients A3 and C3 use the cache of ISP B to download content 3. Clients A1, B1 and C1 use a proximity aware P2P system to exchange content 1, and do not access the cache. The client B4 uses the cache of ISP C to download content 4. All three ISPs save on IP transit traffic.

theoretic terms, from which no ISP has an interest to deviate. We propose two distributed algorithms to solve the cooperative caching game, and use extensive simulations to verify that the algorithms converge to an equilibrium and to evaluate the sensitivity of the potential benefits of cooperation to various parameters. We use trace-driven simulations to quantify the potential benefits of cooperation in terms of the decrease of the ISPs' transit traffic.

The rest of the paper is organized as follows. In Section II we describe the considered cooperative caching scheme and the rationale behind it. Section III presents the game theoretic model of cooperative caching and contains the main analytic results. We describe the distributed algorithms that model cooperative caching in Section IV. We introduce our performance metrics and give bounds on the gains achievable by cooperative caching in Section V, and evaluate the performance gains of cooperative caching in Section VI. Section VII presents the related work, and Section VIII concludes the paper.

II. Background

ISPs ensure global reachability through buying IP transit services and through maintaining bilateral or multilateral settlement-free peering agreements. Settlement-free peering agreements enable ISPs of similar size and geographic coverage to exchange peering traffic freely for mutual benefit as long as the traffic balance satisfies some criteria agreed upon by the parties. Hence, the cost of peering is insensitive to short term fluctuations of the amount of peering traffic (as long as the traffic does not cause congestion).

Transit traffic is, on the contrary to peering traffic, usually charged according to the 95 percent rule (i.e., the client pays for the 95 percentile traffic calculated over a month), and hence increased traffic leads to increased costs. Consequently, ISPs could cut costs through decreasing transit traffic by increasing peering traffic, as long as the peering traffic is kept balanced. This is the rationale of the cooperative caching scheme we consider. Typically, contents have similar popularity at nearby ISPs, so that without cooperative caching the caches of peering ISPs would possibly cache the same contents. Cooperative caching makes use of the correlation of content popularities among ISPs to improve the caching efficiency, even though each ISP aims to minimize its transit traffic with its cache. Cooperation involves exchanging information between the caches, but in contrast to hierarchical caching, there is *no authority to coordinate* the operation. Message exchange between the caches can be based, e.g., on an extension of the Internet Cache Protocol [15].

Cooperative caching allows P2P clients in an ISP to use the cache resources of neighboring ISPs if there is a bilateral peering agreement between the ISPs, as shown in Fig. 1. We will use the term *client* for peers in the P2P overlays to avoid confusion, and use the term *relay* for a cache of an ISP serving clients in a peering ISP. We consider the following scheme for the use of P2P caches. Clients follow their (proximity aware or unaware) P2P protocol to exchange data. Whenever a client would download data through a transit link from an external client, it first contacts the available caches and tries to download the data from one of the available caches. If the data are not available in any of the caches, they will be downloaded from the external client and hence generate transit traffic. Ideally, as an effect of cooperation the sets of contents stored in neighboring caches will be disjoint.

Cooperative caching increases the peering traffic between two peering ISPs, but the increase is not more than the sum of the decrease of the transit traffic of the two ISPs. Increased peering traffic might require higher port speeds at peering points, but the upgrade is typically much cheaper than buying transit capacity, hence the decrease of transit costs compensates for the increase of the peering costs. Cooperative caching also increases the load of the cooperating caches, but as we show at the end of this paper, the increase of the cache load is kept moderate if caching is used together with proximity aware P2P systems, while the gains of cooperation are almost unaffected.

We do not discuss the issue of cache discovery in this paper. Neither do we discuss the compatibility of caches with P2P protocols, e.g., with BitTorrent's optimistic unchoking mechanism. There is ongoing standardization work in the Application Layer Traffic Optimization (ALTO) working group of the IETF [14] on service discovery for P2P systems, P2P caches being one kind of service. BEP-022 specifies an extension of the BitTorrent protocol for service discovery [4]. The architecture proposed in [34] is another example as to how ISP supplied entities can be used to provide information to P2P applications.

III. Cooperative caching as a game

In this section we introduce the notations used throughout the paper, describe a game theoretic interpretation of cooperative caching strategies, and provide the main analytic results of the paper.

A. System model

We model the network of ISPs with a graph $\mathcal{G} = \{I, E\}$. Each vertex of $\mathcal{G}$ corresponds to an ISP and there is an edge $\{i, i'\}$ between vertexes i and i' if the corresponding ISPs have a settlement-free peering agreement. We denote the degree of node $i \in I$ by δ_i, the minimum node degree in $\mathcal{G}$ by δ, and use the notation $\mathcal{P}(i)$ for the set of neighbors of i. We do not model multiple links connecting two ISPs and we assume

I	Set of ISPs
$\mathcal{P}(i)$	Set of peering ISPs of ISP i (neighborhood set)
K_i	Cache capacity installed at ISP i
$\mathcal{H}$	Set of contents
S^h	Size of content h
$N_i^h(t)$	# of peers interested in content h in ISP i at time t
$B_i^h(t)$	Transit traffic related to content h in ISP i at time t *without* caching
$T_i(t)$	Transit traffic in ISP i at time t *with* caching
$r_{i,i'}^h(t)$	Relaying ratio of ISP i to i' for content h at time t
dr^-	Min. speed of change of $r_{i,i}^h(t)$
dr^+	Max. speed of change of $r_{i,i}^h(t)$

TABLE I
LIST OF FREQUENTLY USED NOTATIONS.

that peering capacities are sufficient to carry all relayed traffic, hence we do not consider link capacity constraints.

Let us denote the set of available P2P content (e.g., torrents, or files) by $\mathcal{H}$. The size of content $h \in \mathcal{H}$ will be denoted by S_h. We denote by $N^h(t)$ the number of peers that are interested in content h globally, and by $N_i^h(t)$ their number in ISP i. We denote the IP transit traffic generated by content h in ISP i at time t without caching by $B_i^h(t) = f(N_i^h(t), N^h(t))$. We describe our approximation of the transit traffic in Section VI-B2, but the exact form of $B_i^h(t)$ does not affect the analytical results presented in this section.

Let us denote the cache capacity installed at ISP i by K_i. We describe the relaying strategy of ISP i with respect to content h at time t by the real-valued function $r_{i,i'}^h(t) : \mathcal{H} \times I^2 \times \mathbb{R} \to [0,1]$. Then $r_{i,i}^h(t) = 1$ if content h is cached in ISP i at time t, and $r_{i,i}^h(t) = 0$ if it is not; $0 < r_{i,i}^h(t) < 1$ means that the content is partially cached. Similarly, $r_{i,i'}^h(t) = 1$ corresponds to content h being cached in i and being relayed to i' at time t; $r_{i,i'}^h(t) = 0$ means that the content is not relayed, and $0 < r_{i,i'}^h(t) < 1$ means that the content is partially relayed. By definition $r_{i,i'}^h(t) = 0$ for all $i' \notin \mathcal{P}(i)$ $(i \in I, h \in \mathcal{H})$, and $r_{i,i'}^h(t) \le r_{i,i}^h(t)$. The relaying strategy of every ISP has to satisfy cache capacity constraints, i.e.,

$$\sum_{h \in \mathcal{H}} S^h r_{i,i}^h(t) \le K_i \qquad i \in I. \tag{1}$$

The speed at which contents can be cached depends on the speed at which the contents can be obtained (e.g., from clients in the ISP and its peering ISPs). We will denote the maximum speed (in terms of MB per time unit) at which the caching ratio of contents can be increased and decreased by dr^+ and dr^- respectively (dr^- can be assumed to be $-\infty$ as it corresponds to data being deleted from the cache). Without loss of generality we use $t = 0$ to denote the time instance when a relaying decision has to be made, and will omit the time dimension whenever the context allows it.

Popular second generation P2P file sharing protocols, like Gnutella, FastTrack and BitTorrent divide contents into many small parts and typically obtain the different parts from a number of different clients. The small parts are called *pieces* in BitTorrent terminology, *chunks* in FastTrack and *segments* in Gnutella terminology. We will use the terms piece and chunk interchangibly to refer to parts of contents. The size of the pieces varies depending on the P2P protocol: for BitTorrent the piece size in bytes is a power of two, usually between 16kB and 2048kB; in Gnutella the piece size can be specified as a percentage of the content size or can be dynamically adjusted depending on the connection speed. The piece size is typically small compared to the content's size in order to make the content distribution efficient. For example, results shown in [22] indicate that for BitTorrent the best performance is achieved with about thousand pieces per content item.

Consequently, a P2P cache will have to serve requests for pieces anywhere in the file [13], [32]. If the cache contains a share $r_{i,i}^h(t)$ of content h, then the probability that a request cannot be served from the cache can be expressed as $1 - r_{i,i}^h(t)$.

B. Caching games

In the following we define three caching games: non-cooperative, cooperative with selfish ISPs and cooperative with altruistic ISPs.

Non-cooperative caching: In the case of non-cooperative caching the pieces that are to be downloaded over a transit connection can only be served from the cache installed by the ISP. The remaining transit traffic is the sum of the traffic that cannot be served from the cache over all contents

$$T_i^C(t) = \sum_{h \in \mathcal{H}} (1 - r_{i,i}^h(t)) C_i^h(t) = \sum_{h \in \mathcal{H}} (1 - r_{i,i}^h(t)) B_i^h(t), \tag{2}$$

where the cost function $C_i^h(t)$ equals to the transit traffic without caching $B_i^h(t)$. The goal of ISP i is to minimize $T_i^C(t)$ by adjusting $r_{i,i}^h(t)$. Cache eviction policies that perform this minimization were proposed and evaluated in [13], [32] based on measured traces of FastTrack and Gnutella traffic.

The local cost game (LC): In the local cost game every ISP follows its self-interest, but cooperates with its peering ISPs to minimize its own transit traffic. Consequently, the requests for pieces of the contents can be served from the caches of peering ISPs as well. In our model we assume that the content is the smallest unit at which neighboring caches can coordinate: a cache can know what portion of a content a neighboring cache stores but not which parts its stores. This is a pessimistic assumption, but a reasonable one: given the high amount of content available in P2P networks and the large content sizes, sub-content level coordination would lead to tremendous communication overhead between the caches. As an example, if the caches of two neighboring ISPs i and i' cache 50 percent of content h each, i.e., $r_{i,i}^h(t) = r_{i',i'}^h(t) = 0.5$, then the probability that a request for content h can not be served is $(1 - r_{i,i}^h(t)) \times (1 - r_{i',i'}^h(t)) = 0.25$. If the caches were coordinated on a sub-content level (e.g., byte or block level) then the corresponding probability could be as low as $1 - r_{i,i}^h(t) - r_{i',i'}^h(t) = 0$.

Given our assumption, the probability that a client in ISP i can not download a requested piece of content h from a cache in ISP i or a peering ISP $i' \in \mathcal{P}(i)$ can be expressed as $\prod_{i' \in \{i \cup \mathcal{P}(i)\}} (1 - r_{i',i}^h(t))$. The transit traffic is then the traffic that cannot be served from any of the available caches

$$T_i^{LC}(t) = \sum_{h \in \mathcal{H}} (1 - r_{i,i}^h(t)) C_i^h(t), \tag{3}$$

where the cost function $C_i^h(t)$ is

$$C_i^h(t) = B_i^h(t) \prod_{i' \in \mathcal{P}(i)} (1 - r_{i',i}^h(t)), \tag{4}$$

i.e., the requests that originate in ISP i and can not be served from the caches of the neighboring ISPs of ISP i.

The goal of ISP i is to minimize $T_i^{LC}(t)$ by adjusting $r_{i,i}^h(t)$ as a function of the relaying strategies $r_{i',i}^h(t)$ of the peering ISPs. We refer to this game as the *local cost* game (LC).

The neighborhood cost game (NC): Without any incentives for altruism, selfish behavior is expected from the ISPs. Given the right incentives, ISPs could however show altruistic behavior. Our goal here is to understand how much the ISPs could gain in a system built on altruistic behavior.

In the *NC* game the clients of ISP i are allowed to download from the caches of the peering ISPs just like in the *LC* game. The altruism is introduced by letting every ISP minimize the sum of its own transit traffic and that of its peering ISPs. The amount of transit traffic to be minimized can be expressed as

$$T_i^{NC}(t) = \sum_{h\in\mathcal{H}} (1-r_{i,i}^h(t))C_i^h(t), \tag{5}$$

where the cost function $C_i^h(t)$ is

$$C_i^h(t) = B_i^h(t)\prod_{i'\in\mathcal{P}(i)}(1-r_{i',i}^h) + \sum_{i'\in\mathcal{P}(i)} B_{i'}^h(t)\prod_{i''\in\{\mathcal{P}(i')\setminus i\}}(1-r_{i'',i'}^h), \tag{6}$$

i.e., the requests that originate in ISP i or any of its neighboring ISPs, and can not be served from the caches of the respective neighboring ISPs excluding ISP i.

The goal of ISP i is to minimize $T_i^{NC}(t)$ by adjusting $r_{i,i}^h(t)$ as a function of the relaying strategies $r_{i',i}^h(t)$ of the peering ISPs, and implicitly the peering ISPs of those ISPs. The difference between the cost functions in (4) and (6) is the second term, which corresponds to the sum of the traffic of the peering ISPs i' that cannot be served from any of the caches that are available to them, except for the cache of ISP i. We refer to this game as the *neighborhood cost* game (NC). We note that this game can not be transformed into a *LC* game of neighborhoods each controlling their own resources, because the neighborhoods are overlapping.

C. Nash equilibria in cooperative caching

Given the expressions for the traffic to be minimized, it is not obvious whether a network of cooperative caches can reach a stable cache allocation given a stable distribution of the traffic arriving to the caches in the individual ISPs ($B_i^h(t)$). In the following we show that independent of whether the ISPs are selfish or altruistic, i.e., for both the *LC* and the *NC* game, there is a stable allocation of the caches.

In game theoretic terms we are interested in whether the games defined above have a pure strategy Nash equilibrium: an allocation of cache resources from which no ISP has an interest to deviate given the allocations of the other ISPs. Let us define the relaying vector of ISP i as $\mathbf{r}_i = (r_{i,i}^1,\ldots,r_{i,i}^{|\mathcal{H}|})$. The relaying vector of ISP i lies within the closed ball $\mathcal{B}_i \subset \mathbb{R}^{|\mathcal{H}|}$ defined by $\sum_{h\in\mathcal{H}} S^h r_{i,i}^h(t) \le K_i$ (i.e., all solutions have to satisfy the cache capacity constraint (1)). Furthermore, we define the relaying vector $\mathbf{r}_{-i}$ of all ISPs except ISP i as the Cartesian product of their relaying vectors. The set of feasible relaying vectors $\mathbf{r}_i$ is partially ordered in a componentwise sense, but it is not a lattice because of the cache capacity constraints.

For each ISP i we can define the payoff function, which it aims to maximize

$$f_i(\mathbf{r}_i,\mathbf{r}_{-i}) = -\sum_{h\in\mathcal{H}}(1-r_{i,i}^h)C_i^h,$$

where C_i^h was defined in (4) and (6) for *LC* and *NC* respectively. Using these definitions we formulate the following theorem, which shows the existence of a Nash-equilibrium.

Theorem 1: For both the *LC* and the *NC* strategies, and for a mixture thereof, there exists at least one pure strategy Nash equilibrium, i.e., a set of relaying vectors $\bar{\mathbf{r}}_i$ such that

$$f_i(\bar{\mathbf{r}}_i,\bar{\mathbf{r}}_{-i}) \ge f_i(\mathbf{r}_i,\bar{\mathbf{r}}_{-i}) \quad \text{for } i\in I, \mathbf{r}_i \in \mathcal{B}_i.$$

Proof: The proof of the theorem is based on Kakutani's fixed point theorem and is shown in the Appendix. ∎

D. Optimal cache allocation

We define the optimal cache allocation as the one that minimizes the sum of the transit traffic of the individual ISPs. The corresponding constraint optimization problem is to minimize the total transit traffic of all ISPs subject to the constraints on cache capacity (8) and the maximum and minimum speed of reconfiguration (9). We refer to the optimization problem as the *globally optimal cooperative caching problem* (GCCP).

$$\begin{aligned}
\text{min.} \quad & \sum_{i\in I}\sum_{h\in\mathcal{H}}\int_0^\infty B_i^h(t)\prod_{i'\in\{i\cup\mathcal{P}(i)\}}(1-r_{i',i}^h(t))dt && (7)\\
\text{s.t.:} \quad & \sum_{h\in\mathcal{H}} S^h r_{i,i}^h(t) \le K_i \qquad i\in I && (8)\\
& dr^- < S^h(t)\frac{\partial}{\partial t}r_{i,i'}^h(t) < dr^+ \quad i,i'\in I,\ h\in\mathcal{H} && (9)\\
& r_{i,i}^h(t) \ge r_{i,i'}^h(t) \qquad i\in I,\ i'\in\mathcal{P}(i),\ h\in\mathcal{H} && (10)
\end{aligned}$$

The future relaying strategies have to be chosen based on the current relaying strategies $r_{i,i'}^h(0)$, and the future traffic load of the caches $B_i^h(t)$. Let us call the solution to GCCP the *globally optimal cooperative caching and relaying* strategy (OCR). In practice, calculating and enforcing the OCR strategy in the context of the considered cooperative caching scheme is not feasible because it would require a trusted central authority, nevertheless OCR serves as a comparison to evaluate the efficiency of the Nash equilibria.

IV. Cooperative Caching Algorithms

Caching algorithms considered for P2P traffic gradually replace the least popular contents by the contents that are becoming popular. The popularity of contents is usually estimated based on their request rate. Cache eviction policies that follow this approach for P2P systems were described, e.g., in [13], [32]. The extension of these eviction algorithms to cooperative caching is simple by choosing a suitable definition of the request rate.

In the case of the local cost game (LC) the request rate is the rate of requests that originate in the local ISP and cannot be served from the caches of peering ISPs. Every cache can measure its own request rate, there is no need for information exchange between the cooperating caches. The cooperation can, however, be made more efficient if the neighboring caches exchange information about the contents they cache (i.e., $r^h_{i,i'}$), as it becomes easier for a cache to locate a suitable neighboring cache for the local requests.

In the case of the neighborhood cost game (NC) the request rate is the sum of (i) the rate of requests that originate in the local ISP and cannot be served from the caches of the peering ISPs and (ii) the rate of requests that originate in peering ISPs and cannot be served from any of the caches available to them. In order to implement this algorithm the neighboring caches would have to exchange information about the rate of requests that originate locally and cannot be served from the caches of the peering ISPs. Similar to the LC game, the neighboring caches can exchange information about the contents they cache (i.e., $r^h_{i,i'}$) in order to make it easier for a cache to locate a neighboring cache that can potentially serve its local requests.

A. Algorithms for cooperation

The distributed algorithms we use implement the behavior described above. The algorithms can be used in *on-line* and *off-line* mode. In the on-line mode contents are cached during the iterations of the algorithm according to changing relaying decisions; in the off-line mode contents are cached once the algorithm converges to a solution. We will show results using both modes in Section VI.

The pseudo-code of the *NC* algorithm is shown in Table II. In step (1) the ISP calculates the cost of every content h based on the local requests (B^h_i) not served by neighboring ISPs' caches and based on the costs reported by the neighboring ISPs. In step (2) the ISP finds the content that has the highest cost normalized by its size (h^+) and is not entirely cached. In step (3) the ISP finds the content that has the lowest cost normalized by its size (h^-) and is at least partially cached. If $h^- = h^+$ then the iteration is finished in step (4), otherwise the algorithm has to evict part of content h^- to be able to store more of content h^+.

The rate of change is calculated in step (5) and is limited by two factors: by dr^+ in the on-line mode of operation, and by the reciprocal of the node degree (scaled by a factor ε) in the case of off-line operation. The scaling factor ε influences the communication overhead between caches and the convergence speed. The higher the value of ε, the slower the convergence and the higher the communication overhead, but a high value of ε helps to damp oscillations around an equilibrium state. The rationale for assigning a lower rate of change to nodes with a high degree in off-line mode is to avoid potential instabilities that could be caused by changes made by high degree nodes.

To illustrate the necessity of limiting the rate of change consider an example with two ISPs ($I = \{1,2\}$) with unit cache capacity ($K_i = 1$), and two contents ($\mathcal{H} = \{1,2\}$) of unit size ($S^h = 1$) and $B^1_i > B^2_i$. Let the relaying vectors be initially $\mathbf{r}_i = (1,0)$. If the two ISPs update their relaying vectors simultaneously to minimize their costs given the other ISP's relaying vector then both will change to $\mathbf{r}_i = (0,1)$. A subsequent update will result in $\mathbf{r}_i = (1,0)$, etc. By limiting the rate of change we do not avoid such cycles around the Nash equilibria, but we can decrease their amplitude.

```
ISP $i$ executes every $\tau$ time
1. Calculate for $h \in \mathcal{H}$
   $C^h_i = B^h_i \prod_{i' \in \mathcal{P}(i)} (1 - r^h_{i',i}) + \sum_{i' \in \mathcal{P}(i)} C(i', i, h)$
2. Pick $h^+$ for which $r^{h^+}_{i,i} < 1$ and $C_i(h^+)/S^{h^+}$ is maximal.
3. Pick $h^-$ for which $r^{h^-}_{i,i} > 0$ and $C_i(h^-)/S^{h^-}$ is minimal.
4. If $h^+ = h^-$ then finish iteration.
5. Calculate $\beta$, the allowed rate of change
$\beta = min(dr^+\tau/S^{h^+}, 1/(|\mathcal{P}(i)|+1)/\varepsilon, 1 - r^{h^+}_{i,i}, r^{h^-}_{i,i} S^{h^-}/S^{h^+})$.
6. Set $r^{h^+}_{i,i} = r^{h^+}_{i,i} + \beta$.
7. Set $r^{h^-}_{i,i} = r^{h^-}_{i,i} - \beta S^{h^+}/S^{h^-}$.
8. For every $i' \in \mathcal{P}(i)$ recalculate
   $C(i, i', h^+) = B^h_i \prod_{i'' \in \{i \cup \mathcal{P}(i) \setminus i'\}} (1 - r^{h^+}_{i'',i})$ and
   $C(i, i', h^-) = B^h_i \prod_{i'' \in \{i \cup \mathcal{P}(i) \setminus i'\}} (1 - r^{h^-}_{i'',i})$.
9. Send $r^{h^+}_{i,i'}$, $C(i, i', h^+)$, $r^{h^-}_{i,i'}$, $C(i, i', h^-)$ to ISP $i' \in \mathcal{P}(i)$.
```

TABLE II
PSEUDO CODE OF THE DISTRIBUTED CACHING ALGORITHM FOR THE *NC* GAME.

In steps (6) and (7) the ISP increases the caching ratio of the most expensive content (h^+), and decreases the caching ratio of the least expensive content (h^-) according to the allowed rate of change. In step (8) the ISP recalculates the costs ($C(i,i',h^+)$ and $C(i,i',h^-)$) of the contents that were adjusted in steps (6) and (7) and in step (9) it sends the new costs to the caches of the peering ISPs. These costs will be used by the peering ISPs in step (1) of the algorithm next time they execute it.

We do not show the pseudo code of the *LC* algorithm because it differs from the *NC* algorithm in three points only: (i) in Step (1) $C(i',i,h) = 0$, (ii) Step (8) does not have to be performed and (iii) in Step (9) only $r^{h^+}_{i,i'}$ and $r^{h^-}_{i,i'}$ have to be sent to the neighboring caches.

B. Convergence to equilibria

The convergence of distributed algorithms to Nash equilibria is in general hard to prove. The distributed caching algorithms described above would converge to an equilibrium if there were no cache capacity constraints and if the algorithms were executed according to the Nash dynamics, that is, at each step one ISP switches its relaying strategy to a better alternative. Without cache capacity constraints the set of relaying vectors is a lattice under the componentwise partial ordering, and the convergence in this case follows from the sub-modularity of the traffic cost functions of the *LC* and the *NC* games (eqs. (3) and (5)) [35]. The convergence is however not guaranteed if the algorithms are executed in parallel. Furthermore in the presence of cache capacity constraints the set of relaying vectors is not a lattice. Hence, we use simulations to verify the convergence of the distributed algorithms.

V. Performance measures and bounds

The price of anarchy and the optimistic price of anarchy [7] are often used to evaluate the inefficiency of Nash-equilibria compared to the optimal solution. Since a centralized solution is not feasible for cooperative caching, we are rather interested in the average gains that one can achieve in the equilibria.

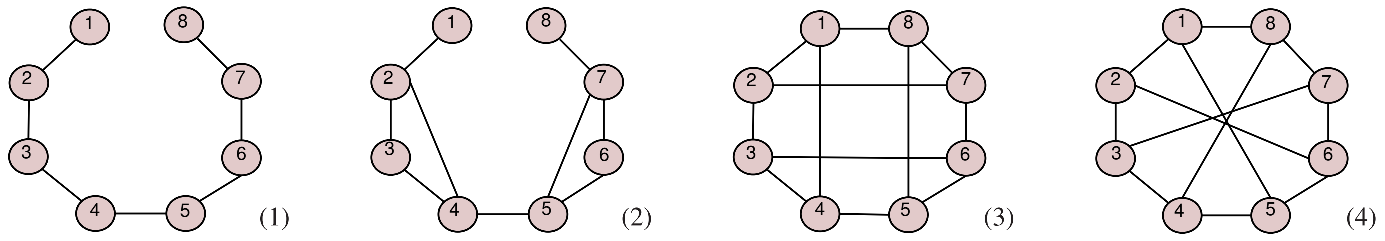

Fig. 2. Graphs (1)-(4): Graph (1): $\delta = 1$, $D_1(\mathcal{G}) = 2$. Graph (2): $\delta = 1$, $D_1(\mathcal{G}) = 2$. Graph (3): $\delta = 3$, $D_1(\mathcal{G}) = 4$. Graph (4): $\delta = 3$, $D_1(\mathcal{G}) = 2$.

Consequently, the measures we use capture the average performance benefits of cooperative compared to non-cooperative caching, both for Nash-equilibria and for the optimal solution.

We define two measures to quantify the gain of cooperation: the peering gain and the traffic gain. We define the peering gain for ISP i as

$$PG_i = \frac{1}{K_i} \sum_{h \in \mathcal{H}} S^h (1 - \prod_{i' \in \{\mathcal{P}(i) \cup i\}} (1 - r^h_{i',i})), \qquad (11)$$

and the mean peering gain as $PG = \frac{1}{|I|} \sum_{i \in I} PG_i$. PG_i quantifies the increase of the available amount of cached content as seen by the clients of ISP i due to cooperation, the higher the better.

Similarly, we define the traffic gain for ISP i as the ratio of the amount of traffic served from caches using cooperative caching and that served from caches using non-cooperative caching,

$$TG_i = \frac{\sum_{h \in \mathcal{H}} B^h_i (1 - \prod_{i' \in \{\mathcal{P}(i) \cup i\}} (1 - r^h_{i',i}))}{\sum_{h \in \mathcal{H}} r^h_{i,i} B^h_i},$$

and the mean traffic gain as $TG = \frac{1}{|I|} \sum_{i \in I} TG_i$. With non-cooperative caching ISP i should install $PG_i K_i$ cache capacity instead of K_i in order to achieve a TG_i fold increase of the traffic served from a cache.

A. *Performance bounds*

In the following we derive lower and upper bounds for the peering gain. Without loss of generality we limit ourselves to the evaluation of relaying strategies on a set of ISPs I connected by peering agreements, i.e., $\mathcal{G}$ is a connected graph. We focus on the case when the contents are equally popular in all ISPs. We argue that this assumption is likely to be valid for ISPs with settlement-free peering agreements as they are typically within the same country or region.

The amount of content that is available (cached or relayed) in any ISP can be bounded by

$$\overline{PG}_i \hat{=} 1 + \frac{1}{K_i} \sum_{i' \in \mathcal{P}(i)} K_{i'} \geq PG_i, \qquad (12)$$

which is proportional to the degree of the ISP. The mean peering gain can be bounded based on (12) by

$$\overline{PG} \hat{=} 1 + \frac{1}{|I|} \sum_{i \in I} \frac{\sum_{i' \in \mathcal{P}(i)} K_{i'}}{K_i} \geq \frac{1}{|I|} \sum_{i \in I} PG_i = PG. \qquad (13)$$

Both $\overline{PG}_i$ and $\overline{PG}$ are only dependent on the graph topology and the cache capacities. $PG_i = \overline{PG}_i > 1$ means that there is no overlap in the cached contents in ISP i and ISPs $i' \in \mathcal{P}(i)$.

If the amount of cache capacity is equal in all ISPs ($K_i = K$) then we can also obtain a lower bound on the efficiency of cooperative caching for the optimal allocation *OCR*.

Lemma 1: For an arbitrary connected graph $\mathcal{G}$ and equal cache capacities in the ISPs, in OCR the peering gain of every ISP is bounded from below by

$$PG_i \geq D_1(\mathcal{G}) \geq 2. \qquad (14)$$

Proof: In order to obtain a worst case lower bound on the peering gain we make the following observation. The worst case scenario for cooperative caching is if the traffic cost of the k^{th} popular content is infinitely higher than that of the $k+1^{st}$ most popular content for all k. In this case all ISPs are interested in caching only the most popular contents. Hence finding OCR in the worst case is closely related to finding minimum dominating subsets of I, a well-studied problem in graph theory. For $K_i = 1$ ($i \in I$) finding OCR is related to finding the domatic number $D_1(\mathcal{G})$ of graph $\mathcal{G}$, i.e., the maximum number of disjoint dominating subsets of I [11]. For $K_i = K \geq 1$ ($i \in I$) the problem is known as finding the r-configuration $D_r(\mathcal{G})$ of graph $\mathcal{G}$ [11].

For any connected graph $D_1(\mathcal{G}) \geq 2$. Furthermore, for the r-configuration of a graph $D_r(\mathcal{G}) \geq r D_1(\mathcal{G})$ [11]. The proof of the lemma then follows from the definition of PG_i. ■
Consequently, given an optimal resource allocation, ISPs can at least double the amount of cached contents and hence eventually halve the IP transit traffic through cooperative caching compared to non-cooperative caching if all of them deploy the same amount of cache resources. Alternatively, it is enough for them to install half as much cache capacity as in the case of non-cooperative caching.

VI. Performance evaluation

We developed a distributed simulator to evaluate the behavior of the considered cooperative caching algorithms. In the simulator the several nodes execute the cooperative caching algorithms in parallel, as caches would update their contents in parallel. Unless otherwise stated, we start the simulations from the optimal non-cooperative cache allocations and run the simulations until the results converge. The results shown are the averages of 10 runs of the algorithms, the results are within a 1 percent interval at a 95 percent level of confidence.

We use the bounds developed in the previous section as a reference to evaluate the efficiency of the considered *LC* and *NC* games. As an additional reference we use a multi-population genetic algorithm (*GA*) with 20 subpopulations to solve the GCCP. In order to help the genetic algorithm, we

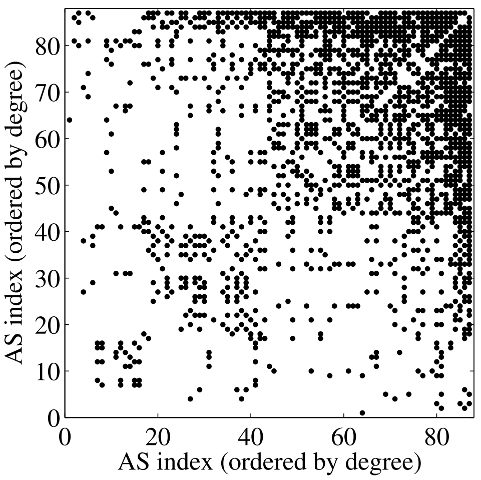

Fig. 3. Graph ASP: Adjacency matrix of 87 ASs in Northern Europe.

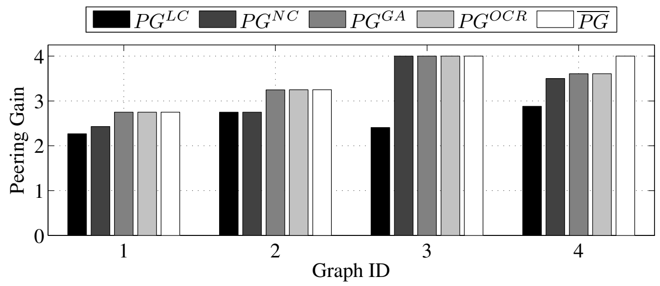

Fig. 4. Average peering gain achieved using various algorithms and the theoretical upper bound for $K_i = 1$ on graphs (1)-(4).

place one allocation provided by the *LC* and the *NC* games in each of the 20 populations.

We use various ISP peering topologies for the evaluation.

Toy topologies: Graphs (1)-(4) are shown in Fig. 2. The number of ISPs is $|I| = 8$ in these graphs, but the graphs have loops of different lengths and differ in their domatic numbers. While these graphs do not represent real ISP topologies, their simplicity makes it possible to understand the operation of the considered strategies.

AS level peering topology: We obtained the graph of the settlement-free peering agreements between 87 autonomous systems (ASs) in Northern Europe (Denmark, Finland, Norway and Sweden) from the BGP route advertisements of the ASs stored in the RIPE *whois* database. We considered the advertisements that correspond to bilateral peering relations only. We identified a bilateral peering relation by both ASs advertising only their own AS number to each other, and a transit relation by one of the ASs advertising *"any"* to the other. We refer to this graph as Graph ASP. Fig. 3 shows a graphical representation, in which a dot stands for an edge between two nodes, of the adjacency matrix of Graph ASP. The minimum node degree in the graph is $\delta = 1$, consequently, the domatic number of the graph is $D_1(\mathcal{G}) \leq 2$, but the maximum node degree is 62 and the upper bound of the average peering gain is $\overline{PG} = 19.48$. The nonlinear minimum least squares fit for a Zipf distribution to the degree-rank statistics of the graph is $75.67k^{-0.42}$, with root mean squared error 4.68. We observe a dense subgraph consisting of about 20-40 ASs well connected to each other (upper right corner), and the rest of the ASs are also connected to at least some ASs with high degree. In reality several ASs might belong to the same ISP, but we will use AS and ISP interchangibly in the rest of the paper for simplicity.

Random graphs: In addition to the above five topologies, we use random graphs with different topological properties. Details about the random graphs are given in the respective sections.

A. Evaluation using synthetic popularity distributions

In this subsection we show results obtained with synthetic popularity distributions on various graph topologies. Since the solutions given by *LC* and *NC* depend on the distribution of B_i^h/S^h, we fix $S^h = 1$ for Section VI-A and will change the distribution of B_i^h only.

We start the evaluation with Graphs (1)-(4). We set the number of subscribers equal in all ISPs, the total client population is 10^6 out of which 10^5 are within the considered ISPs, distributed uniformly among ISPs, and let the traffic generated by contents, B_i^h, follow a Zipf distribution with exponent $\alpha = 0.7$ [13]. Fig. 4 shows the achieved mean peering gain for the *LC* and the *NC* games on Graphs (1)-(4). As a comparison we show the mean peering gain achieved by the multi-population genetic algorithm (*GA*), the optimal solution (*OCR*), and the upper bound $\overline{PG}$. The figure shows that the gains in the *NC* game are always at least as high as in the *LC* game, and significantly higher in the case of Graph (3), which one would expect to be the most straightforward topology. For Graph (3) *LC* does not converge to the optimal solution, but simulations show that it does not diverge from it, if started there. In general, both the *LC* and the *NC* games provide however close to optimal results. The genetic algorithm manages to find the optimal solution for all four graphs. Since for large graphs we were not able to obtain the optimal solution, we will use the genetic algorithm as a benchmark for Graph ASP and the random graphs in the rest of the paper.

In the following we present results based on Graph ASP unless otherwise stated. We were not able to calculate the optimal solution *OCR* for Graph ASP, hence we will only use the genetic algorithm as reference. We start the evaluation by considering the same distribution of the user populations and the content popularities as for Fig. 4. Fig. 5 shows the maximum peering gain achievable by the ASs, the peering gains achieved by the *LC* and the *NC* games, and the solution obtained by the *GA* algorithm. We ordered the ASs according to their node degrees in order to make the figure easier to read. The results obtained for the *NC* game and those of the *GA* algorithm are quite close to each other, while those obtained for *LC* lie below. The high gains in the *NC* game compared to the *LC* game should provide an incentive for ASs to follow this slightly altruistic strategy for cooperative caching.

The peering gains of the ASs with low node degrees achieve their upper bounds ($\overline{PG}_i = \delta_i + 1$), it is the nodes with node

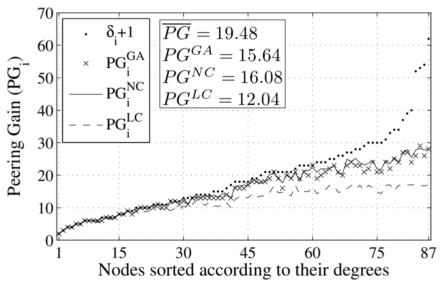

Fig. 5. Peering gains and the theoretical upper bound for $K_i = 1$ on Graph ASP.

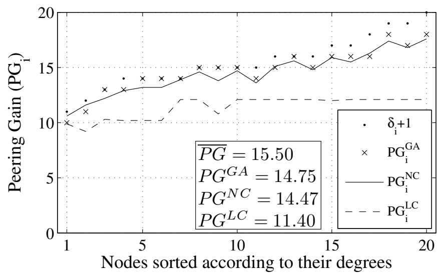

Fig. 6. Peering gains for the best connected 20 ASs in Graph ASP for $K_i = 1$.

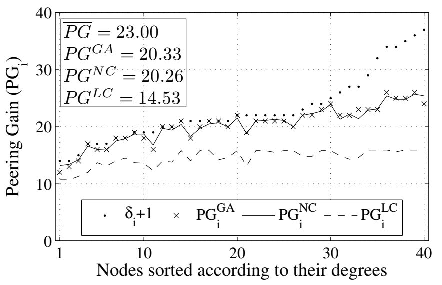

Fig. 7. Peering gains for the best connected 40 ASs in Graph ASP for $K_i = 1$.

degrees above average whos peering gains lie well below their upper bounds. We also note that even though the peering gain obtained in the *NC* game is higher than that of *GA*, the traffic gain obtained by *GA* is higher: the traffic gains are 4.06, 4.41 and 4.45 for *LC*, for *NC* and for *GA* respectively.

1) Incremental deployment: One would suspect that the peering gains of the ASs with high node degrees are negatively affected by the ASs with low node degrees. This is however not true. Figs. 6 and 7 show the peering gains that the best connected 20 and 40 ASs could achieve if they were not peering with the worse connected ASs (i.e., the figures show the peering gain in the dense subgraphs of Graph ASP with 20 and 40 nodes respectively). From (12) we know that the upper bound $\overline{PG}_i$ of the peering gain of an AS decreases if any of its peers is removed. The figures show that the actual peering gains are slightly lower as well. The top 20 ASs achieve a lower average peering gain (*PG*) than when all ASs cooperate. Comparing the average peering gains in Figs. 5 and 7 it might seem that the top 40 ASs benefit from not cooperating with the worse connected ASs, the comparison is however misleading. The average peering gain for these 40 nodes would be 15.72, 22.75 and 22.1 for *LC*, *NC* and the *GA* algorithm if they cooperated with the worse connected ASs (as they did in Fig. 5). The average traffic gains also show the benefits of increased cooperation: for 20 ASs the traffic gains are 4.19, 4.50 and 4.55 for *LC*, *NC* and *GA* respectively (Fig. 6); for 40 ASs the traffic gains are 4.67, 5.17 and 5.21 for *LC*, *NC* and *GA* respectively (Fig. 7). Hence, there is an incentive for ASs to establish peering relationships and cooperative caching with as many other ASs as possible.

2) Does the AS degree distribution matter?: In general it is difficult to discover peering relations between ASs [25], and there is no clear understanding of the distribution of the number of peering agreements of ASs. Hence we consider two models to construct random graphs: the Erdős-Rényi model (ER), and the Barabási-Albert model (BA) [2]. In graphs generated using the ER model the degree distribution of the ASs follows a binomial distribution. In graphs generated using the BA model the AS degree distribution follows a power-law. To make the results comparable to those obtained with Graph ASP all random graphs have 87 nodes.

Fig. 8 shows the average peering gain as a function of the average AS peering degree for the two kinds of random graphs. Surprisingly, the *LC* game yields better results on *BA* graphs, while the *NC* game yields better results on ER graphs: node degrees are more homogeneous in ER graphs, and as we observed on Graph (3), selfish behavior (*LC*) leads to inefficiency on homogeneous graphs with difficult topologies. Altruistic behavior (*NC*) can however benefit from homogeneity. For sparse graphs the results are rather similar for ER and BA graphs. The peering gain for the *LC* game is fairly insensitive to the degree distribution, and for the *NC* game we only observe a significant difference for very dense graphs.

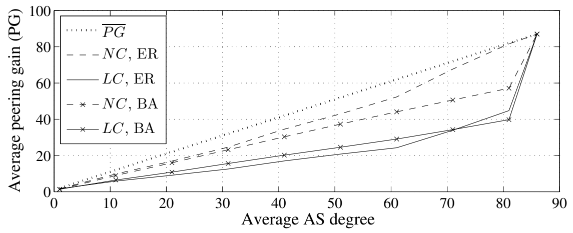

Fig. 8. Peering gain vs. average AS peering degree on ER and BA random graphs, $K_i = 1$.

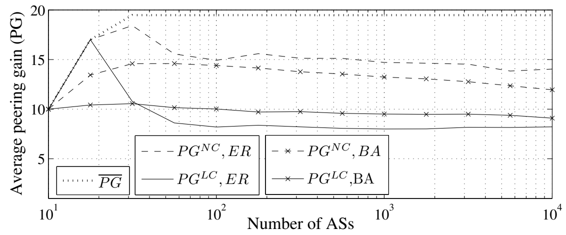

Fig. 9. Peering gain vs. number of ASs on ER and BA random graphs with average degree 18.48, $K_i = 1$.

The results for the *LC* game improve drastically as the graphs become complete. On the complete graphs with 87 nodes the average peering gain is $PG = 87 = \overline{PG}$ for both *LC* and *NC*, and the traffic gains are $TG = 9.59$ and 9.6 respectively. This shows that cooperative caching on a non-complete graph, i.e., the problem considered in this paper, is algorithmically more difficult and yields lower gains than cooperative caching on a complete graph, which is generally considered for cooperative web proxy caching, e.g., [33].

Another important question is how the peering gain scales with the number of ASs if the average peering degree is kept constant, i.e., how would cooperative caching scale to a network of thousands of ASs? We generated random ER and BA graphs of different sizes with the same average AS peering degree as that of Graph ASP, i.e., 18.48. Hence, as the number of ASs grows, the graphs become increasingly sparse. Fig. 9 shows the average peering gain as a function of the

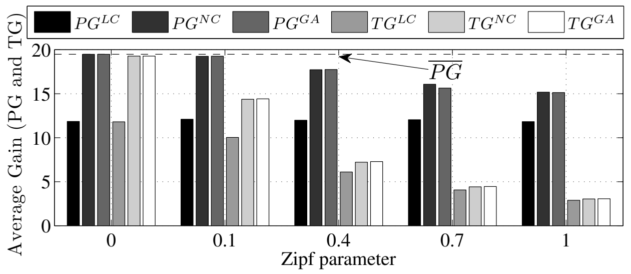

Fig. 10. Sensitivity of the peering gain and the traffic gain to the popularity distribution, $K_i = 1$ on Graph ASP.

number of ASs. The graphs with few nodes are complete or nearly complete, hence the good performance of both the *LC* and *NC* games (as observed in Fig. 8). For graphs larger than about 100 nodes the results are however almost independent of the graph size. This suggests that the results obtained with Graph ASP that consists of 87 ASs with average degree 18.48 are representative for larger graphs with the same average node degree.

3) Sensitivity to the popularity distribution: Most measurements of P2P traffic report a Zipf like distribution of content popularity [13], [32], but the exponent of the distribution varies to some extent. In general, the lower the value of the Zipf exponent, the heavier is the tail of the distribution, and consequently non-cooperative caching is less efficient. Hence it is interesting to see how the efficiency of cooperative caching depends on the tail of the content popularity distribution. Fig. 10 shows the average peering gain and the average traffic gain achieved for different values of the exponent of the Zipfian content popularity distribution. The average peering gain for the *LC* game is almost insensitive to the Zipf exponent, because *LC* does not lead to close to optimal solutions when the popularity distribution is close to uniform (low values of the Zipf exponent). The *NC* game leads however to close to optimal solutions in all cases, hence the peering gain increases as the Zipf exponent decreases. This means that the gains of cooperative caching increase as the tail of the population distribution becomes heavier, i.e., when non-cooperative caching would be less efficient. The traffic gain decreases of course for both games as the tail of the content popularity distribution becomes lighter (i.e., the Zipf exponent increases).

At the two extremes of the parameter space of the popularity distribution we find two well known problems from graph theory. For uniform popularity distribution ($\alpha = 0$) *GCCP* is equivalent to finding disjoint sets of contents for every neighboring ISP, similar to the problem of vertex coloring, which is NP-hard if the number of contents is small [18]. At the other extreme ($\alpha = \infty$), when every content is infinitely more popular than the next popular one, solving *GCCP* involves finding the *r*-configuration of the underlying graph (since the most popular contents must be relayed to every ISP), and is NP-complete [11].

4) Scaling of the peer population: Figure 11 shows the sensitivity of the peering gain and the traffic gain to the population size, the cache capacity and the number of contents for $\alpha = 0.7$. As expected, the results are insensitive to the population size: we do not observe any difference between the results obtained for $N = 10^6$ and for $N = 10^7$. Increasing the number of contents from $|\mathcal{H}| = 200$ to $|\mathcal{H}| = 500$ does not change the results either. The average peering gain is insensitive to doubling the cache capacity from $K_i = 1$ to $K_i = 2$, as well as when the cache capacity is increased proportionally to the number of the contents $|\mathcal{H}|$. The traffic gain decreases of course as the cache capacity increases, because of the decreasing popularity of the cached contents.

5) Convergence to equilibrium: Fig. 12 shows the convergence of the peering and the traffic gain for the *LC* and for the *NC* games as a function of the number of iterations per node for different values of ε. The convergence for the *NC* game is slightly slower than that for the *LC* game. In both cases, the algorithms converge without significant oscillations, after about 70 iterations for $\varepsilon = 1$. The number of iterations needed per node is proportional to $\varepsilon(\overline{PG} - 1)E[K]/E[S]$ and is dominated by the maximum node degree. Nevertheless, it does not depend on ε which equilibrium state is reached. The same observations hold for the convergence of the traffic gain.

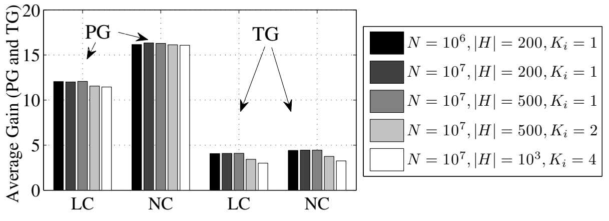

Fig. 11. Sensitivity of the peering gain and the traffic gain to the population size, the cache capacity and the number of contents on Graph ASP.

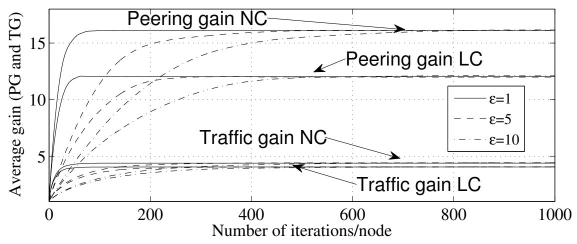

Fig. 12. Convergence for the *LC* and *NC* games for $K_i = 1$ on Graph ASP, peering gain and traffic gain.

B. Evaluation based on measured traces

We use the BitTorrent traces in the Delft BitTorrent Dataset 2 (DBD2) in order to evaluate the efficiency of the cooperative caching scheme with heterogeneous content popularities and content sizes. The subsection starts with a description of the DBD2 data set, followed by the description of the traffic model we use, and it ends with results from trace driven simulations performed with the DBD2 data set.

1) Measurement data set: The DBD2 dataset was collected as part of the MultiProbe project [23]. The traces contain the anonymized IP addresses of the clients that participated in the distribution of the 1916 most popular torrents over a 96 hour period in 2005. Fig. 13 shows the number of clients as a function of the torrent rank in terms of popularity on May 9 2005, 16:20:00 UTC, and exhibits similar characteristics to the data reported in [12], [13]. The figure also shows the non-linear least squares fit for the Zipf distribution to the data, with

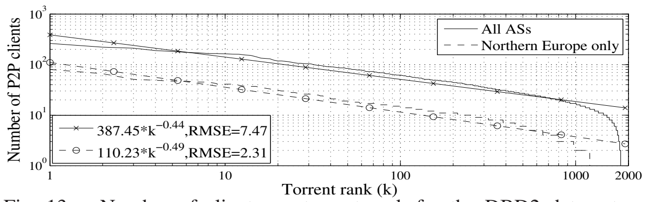

Fig. 13. Number of clients vs. torrent rank for the DBD2 data set on May 9 2005, 16:20:00 UTC.

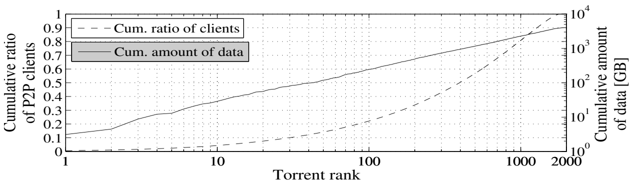

Fig. 14. Cumulative ratio of clients vs. torrent rank and the corresponding amount of data for the DBD2 data set.

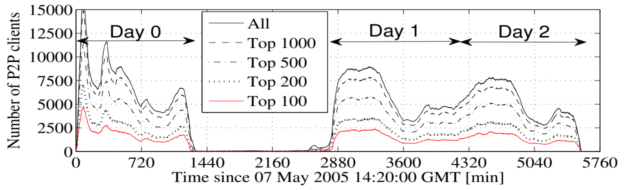

Fig. 15. Number of clients in Northern Europe vs. time for various sets of torrents.

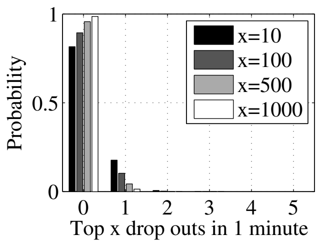

Fig. 16. Dropout from top X over 1 minute.

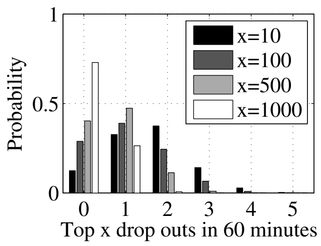

Fig. 17. Dropout from top X over 1 hour.

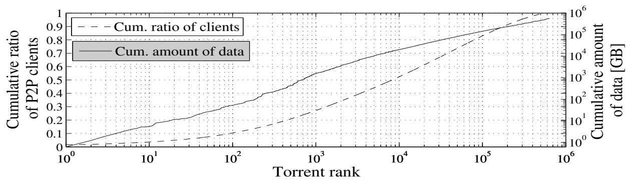

Fig. 18. Cumulative ratio of clients vs. torrent rank and the corresponding amount of data on *mininova.org*.

the corresponding root mean squared errors. Fig. 14 shows the cumulative ratio of the clients as a function of the torrent rank, and the cumulative amount of data as a function of the torrent rank (i.e., the amount of data that is shared in the x most popular torrents). The correlation coefficient between torrent popularity (N^h) and content size (S^h) is 0.12, and shows almost no correlation. As an example, 200 GB of cache would suffice to cache the torrents that 20 percent of the clients belong to, but one would need 800 GB of cache capacity to cache the torrents that 40 percent of the clients belong to. (This is in accordance with results reported for Gnutella traffic in [13].)

We mapped the IP addresses of the clients to the IP addresses allocated to the 87 ASs of Graph ASP in order to obtain the popularity distribution of the 1916 torrents in the different ASs. We identified 903212 BitTorrent clients in the dataset, out of which 138492 are within the considered 87 ASs. Fig. 15 shows the number of concurrent clients in the 87 ASs that participate in the top 100, 200, 500 and 1000 torrents (in terms of number of unique IP adresses) as a function of time over the considered time interval. While the number of clients fluctuates considerably over time, the rankings of the torrents appear to be rather static as shown in Figs. 16 and 17. The figures show the number of torrents that drop out of the top x ($x = 10$, 100, 500 and 1000) in Northern Europe over 1 minute and 1 hour respectively. We found that the dropout is rather small: in the course of one hour on average 1.63, 1.12, 0.73 and 0.28 torrents fall out of the top 10, 100, 500 and 1000 respectively. That is, the higher the number of torrents observed the lower the dropout. Consequently, cooperative caching could eventually lead to a decrease of cache replacements as it increases the amount of cached content as seen by the individual ASs.

The DBD2 data set covers a small subset of the vast amount of contents available on the Internet. In order to verify that the statistical properties of the torrent popularity distribution are representative, on 16 Apr. 2008 we performed a screen-scrape of *www.mininova.org*, the biggest torrent search engine, and collected information about the number of seeds, leechers and the amount of data for each of the 639631 registered torrents. Fig. 18 shows the cumulative ratio of the clients as a function of the torrent rank, and the cumulative amount of data as a function of the torrent rank. The non-linear least squares fit for the Zipf distribution to the popularity-rank statistics is $13364k^{-0.76}$ with root mean squared error 97.49, i.e., the tail of the distribution is lighter than in the DBD2 data set. The number of torrents is almost three orders of magnitude higher for the *mininova.org* data set, but the correlation coefficient between torrent popularity (N^h) and content size (S^h) is similarly small, 0.0135, as for DBD2. Hence, for *mininova.org*, 350 GB of cache would suffice to cache the torrents that 20 percent of the clients belong to, but one would need 8.2 TB of cache capacity to cache the torrents that 40 percent of the clients belong to.

2) Traffic load model: In the following we describe the traffic load model we use to estimate the traffic arriving to the caches based on the $N_i^h(t)$ obtained from the DBD2 data set. If we consider a cache and locality aware P2P system then content would be downloaded from clients in the same AS as first choice, as second choice from the P2P cache or from clients in a peering AS and as a last choice from clients in non-peering ASs. Content downloaded from clients in non-peering ASs generates transit traffic and potentially costs. Let us consider $N_i^h(t)$ clients participating in the distribution of content h in ISP i. If we assume that the cache is only used if the content is not available at a known client in the local AS or in a peering AS, then the request rate arriving to the cache is proportional to

$$B_i^h(t) = N_i^h(t)(1 - l_i^h(t)), \tag{15}$$

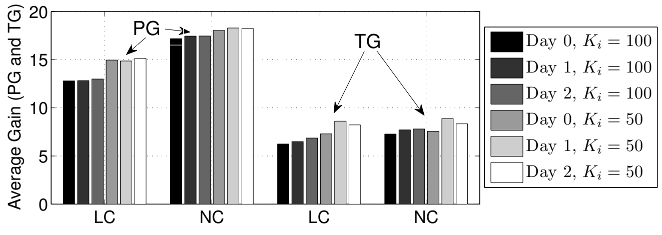

Fig. 19. Average peering gain and traffic gain for 3 days in the DBD2 dataset on Graph ASP. $K_i = 1$

where $l_i^h(t)$ is the proximity awareness factor, which shows how much a client prefers to exchange data with nearby clients. A value of $l_i^h(t) = (N_i^h + \sum_{i' \in \mathcal{P}(i)} N_{i'}^h)/N^h$ corresponds to a proximity unaware P2P system, $l_i^h(t) = 1$ corresponds to a P2P system that only downloads data from clients in the same or in peering ASs. This parameter corresponds to the locality parameter used in [20], and the formula expresses the linear relationship between the proximity awareness factor and the amount of transit traffic shown there. Clearly, this formula does not capture a number of properties of the peer selection process of popular P2P systems (e.g., the optimistic unchoking in BitTorrent) but it is a reasonable approximation of the transit traffic load in an average sense.

3) Daily average gain: In the following we use the popularity distributions $N_i^h(t)$ obtained by mapping the IP addresses in the DBD2 dataset to the 87 ASs of Graph ASP. We set S^h according to the measured torrent sizes. To calculate the daily average gain we calculate the popularity N_i^h of content h in ISP i as the average number of concurrent peers observed in ISP i over a 24 hour period (i.e., average of $N_i^h(t)$ for days 0, 1 and 2), and we consider a proximity unaware P2P system. This definition of popularity corresponds to the classical *most frequently used* eviction policy. We consider two cache capacity sizes, $K_i = 50$GB and $K_i = 100$GB. Proportional to the total amount of data that the DBD2 dataset represents, these cache sizes would be equivalent to approximately 500GB and 1TB respectively in the case of the *mininova.org* dataset. Fig. 19 shows the average peering gain and the average traffic gain achieved based on the popularity distributions for Day 0, 1 and 2 of the DBD2 dataset. We do not observe significant difference between the gains achieved for the different days (of course, the set of cached contents differ to some extent as we will see later). Both the average peering gains and the average traffic gains are comparable to those obtained with the Zipf distribution with exponent 0.7 (for $K_i = 100$GB), and with exponent 0.4 (for $K_i = 50$GB). The reason for the different results for different cache capacities is the change of the slope of the popularity-rank statistics observable above rank 20 in Fig. 13. The high values of the peering gain indicate that the popularity distributions in the peering ASs are rather similar in the DBD2 trace.

The peering and the traffic gains of the various ASs depend on the AS's node degrees in the peering graph and the content popularities in the ASs. In order to quantify how balanced the gains of the different ASs are we define the relaying balance

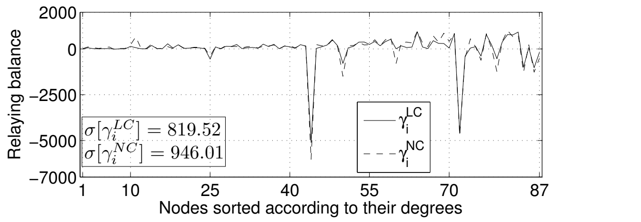

Fig. 20. Relaying balance of the ASs on Day 0 of the DBD2 dataset. $K_i = 1$

between ASs i and i' as

$$\gamma_{i,i'} = \sum_{h \in \mathcal{H}} \int_{-\infty}^{0} r_{i,i'}^h(t) B_{i'}^h(t) - r_{i',i}^h(t) B_i^h(t) dt. \qquad (16)$$

A negative relaying balance between ASs i and i' indicates that the peers in AS i request more traffic from the cache of AS i' than the peers in AS i' from the cache in AS i, that is, AS i is a net receiver. Fig. 20 shows the sum of the relaying balances of every AS, i.e., $\sum_{i' \in \mathcal{P}(i)} \gamma_{i,i'}$, based on the solutions achieved in the *LC* and *NC* games for Day 0. We observe that most ASs have a balance near 0, with the exception of a few outliers. The ASs with negative balance are net receivers, i.e., ASs that receive more content relayed than what they relay to their peers. These ASs can improve their balances by installing more cache resources (though in this case their peering gain would decrease due to the increase of the denominator in (11)) or by establishing more peering relations. The standard deviations of the balances shown in the figure indicate that the *LC* game leads to a better balance of the relaying traffic between the ISPs than the *NC* game.

4) Daily instantaneous gain: Finally, we evaluate the performance of cooperative caching in a dynamic environment. We are interested in how fast the two distributed algorithms can reconfigure the caches, how much data has to be cached (loaded in the caches) due to the reconfiguration, and how the peering gain and the traffic gain are affected during the reconfiguration. For the evaluation we consider the following mode of operation. The popularity distribution in an AS is given by the average request rate $B_i^h(t)$ as calculated in (15) over 24 hours. Every AS updates the statistics every 24 hours, and the updated statistics are used for the execution of the cooperative caching strategies in the on-line mode of operation. Better performance could be achieved by incorporating prediction techniques, but by considering a simple way of operation we can give a pessimistic estimate of the performance. For our evaluation we start the caches from a solution based on the statistics of Day 0 at 2pm on May 10th 2005. We use the statistics from Day 1 as the updated popularity distribution and observe how the caching strategies converge to a new solution by the end of Day 2. The maximum rate at which data can be loaded to the caches is $dr^+ = 0.5$MB/s, while data deletion is immediate, i.e., $dr^- = -\infty$. We calculate the average traffic gain based on the concurrent number of clients interested in the individual torrents $N_i^h(t)$. We consider both proximity unaware and proximity aware systems.

Proximity unaware P2P systems: Fig. 21 shows the average peering gain and the average traffic gain as a function of time for $K_i = 50$GB. The average peering gain remains

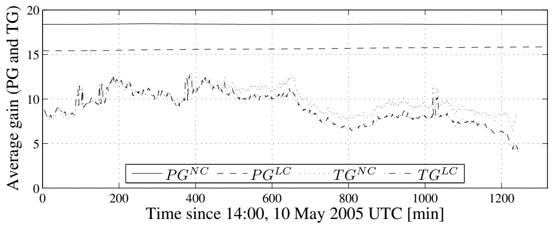

Fig. 21. Average peering gain and traffic gain vs. time for the DBD2 dataset on Graph ASP. $K_i = 50$GB. Proximity unaware P2P system.

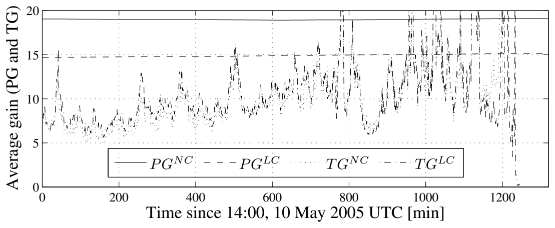

Fig. 22. Average peering gain and traffic gain vs. time for the DBD2 dataset on Graph ASP. $K_i = 50$GB. Ideal proximity aware P2P system.

almost unchanged during the observed interval, the average traffic gain shows however modest fluctuations due to the changing number of clients. Still it remains around the value that we obtained for the static scenarios, and hence, indicates that the considered cooperation strategies can cope with the dynamics of the P2P content of the DBD2 dataset. During the observed time interval in the *LC* game the caches loaded on average 11.2GB of data per AS, while in the *NC* game they loaded 16.6GB of data, which shows that the *LC* game leads to less reconfigurations. Without cooperative caching 24.9GB of data should have been loaded per AS. This result confirms that cooperative caching can not only increase the number of contents that are available through a cache in an ISP but it can also decrease the amount of data that has to be loaded in the caches. This result matches our observation in Section VI-B1 about the decrease of the dropout for large sets of torrents.

Proximity aware P2P systems: In order to show how cooperative caching can complement proximity awareness, we performed the same simulations as for proximity unaware systems but assuming an ideal proximity aware P2P system, which as much as possible, downloads contents from peers in the same ISP or in a peering ISP. In this ideal scheme the locality factor is

$$l_i^h(t) = min(N_i^h(t) - 1 + \sum_{i' \in \mathcal{P}(i)} N_{i'}^h(t), 4)/4, \tag{17}$$

that is, a client in ISP i does not generate transit traffic if there are at least 4 other clients in ISP i or its neighbors. In our simulations proximity awareness decreased the traffic served from the caches by 30 percent on average, because the most popular contents did not need caching. This decrease is in accordance with the ratio of P2P clients found to be 0 or 1 AS hops away in [6].

The gains of cooperation are however not affected by proximity awareness, as shown in Fig. 22. The gains fluctuate significantly more over time than without proximity awareness, because it is the moderately popular contents that are cached. The number of concurrent peers $N_i^h(t)$ varies faster for such contents, which mainly affects the efficiency of non-cooperative caching. We conclude that proximity awareness and cooperative caching can complement each other. On the one hand, proximity awareness decreases the load of the caches. On the other hand, cooperative caching increases the load of the individual caches because of the increased user population.

VII. Related work

Cooperative content caching schemes were first considered for HTTP traffic. Most of the work on caching focused on hierarchical proxy caching strategies, e.g., [10], [28]. The term cooperative caches was used in [31] for hierarchical caches of metadata, but still the approach used a central repository of metadata information.

The idea of adaptive, self-organizing caches was discussed in [29], but the focus of the paper was on how caches could group themselves, and how they could share content information, not on how the caches could adaptively change the content they cache to maximize cache efficiency. The focus of our paper is on the latter, and is hence complementary to [29]. In [33] the authors estimated the gains of cooperative web proxy caching via trace-driven simulations, and a simple analytical model. Similar to other works on cooperative web caching, the evaluation assumes that all caches can cooperate with each other, which would correspond to a complete AS graph for the problem studied in our paper. As we showed in Section VI-A the results are substantially different. Our work differs from previous work on cooperative proxy caching in that we consider the case of partial caching, which was not considered before because of the typically small size of web contents.

Closest to our work in the literature on distributed caching are [7], [19]. In [7] the authors use game theory to study selfish caching of content. Their model differs substantially from ours on several points that affect the properties of the game: it does not consider capacity constraints and allows objects to be accessed at arbitrary distances. Furthermore, the evaluation is based on Nash dynamics protocols, which are not realistic for our cooperative caching problem. In [19] the authors present a game theoretic model of replication on a complete graph with homogeneous distances and binary replication values. The results presented there cannot be generalized to continous replication values and non-complete graphs. Furthermore, as our results show, the results on a complete graph are substantially different from those on sparse graphs.

Related to our work are the studies that consider the efficiency of caching for P2P content distribution. Several measurement studies [17], [13], [32] considered the caching of content for P2P file sharing and showed its possible benefits in decreasing ISP traffic costs. In [30] the authors proposed an application layer protocol that could use existing HTTP caches to decrease the inter-ISP P2P traffic. Finally, cooperation between caches was considered for P2P streaming traffic in [8], but the problem formulation and the solution approach is different from the one considered in this paper.

Related to our work, but different in nature are recent works on ISP friendly content distribution that involve some modification of the P2P application layer protocols. In [1] the authors proposed the introduction of ISP managed oracle nodes that clients can consult in order to obtain a ranking of their neighbors with respect to proximity. Ongoing work in the DCIA P4P working group relies on application layer trackers that allocate bandwidth to P2P applications in order to control traffic related costs [34]. The cooperative caching scheme considered in this paper could be integrated with the above proposals and could lead to improved application performance and decreased costs for ISPs.

VIII. Conclusion

In this paper we have studied whether a cooperative caching scheme could help ISPs to decrease their bandwidth costs caused by peer-to-peer content distribution systems. We gave a game theoretic formulation of the interaction between the caches for two kinds of ISP behavior: selfish and altruistic. We showed the existence of pure strategy Nash equilibria for both games, and gave bounds on the benefits of cooperative caching based on results from graph theory. We evaluated the possible gains of cooperation on diverse graph topologies. Our results show that the gains of cooperation are high even if ISPs follow a selfish strategy (*LC*), but altruistic behavior (*NC*) can further increase the gains of cooperation. Though it provides less gains, the selfish strategy leads to a better balance of relayed traffic and to less reconfigurations of the cache contents. We found that cooperative caching gives incentives for ISPs to establish peering relations, as the gain achievable by an ISP increases with its degree. A major advantage of cooperative caching is that gains are highest when efficient caching is most difficult, i.e., when the tail of the content popularity distribution is heaviest.

We evaluated the efficiency of cooperation on a real AS topology based on a measured trace of BitTorrent content popularity, and conclude that the heterogeneity of content popularities does not affect the performance significantly. We have shown the gains of cooperative caching as content popularity distributions change over time. Our results show that cooperative caching could lead to a significant increase in cache efficiency also in the case of proximity-aware peer selection policies, and hence to a decrease of ISP costs induced by peer-to-peer content distribution systems.

Acknowledgment

The author would like to thank Paweł Garbacki for his help in interpreting the traces in the DBD2 data set. Part of this work was done while visiting the Swedish Institute of Computer Science (SICS).

References

[1] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P users cooperate for improved performance? *ACM SIGCOMM Computer Communication Review*, 37(3), 2007.

[2] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys*, 74(1):47–97, 2002.

[3] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in Bittorrent via biased neighbor selection. In *Proc. of ICDCS*, July 2006.

[4] BitTorrent Local Tracker Discovery Protocol. http://bittorrent.org/beps/bep_0022.html.

[5] CacheLogic. http://www.cachelogic.com.

[6] D. Choffnes and F. Bustamante. Taming the torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In *Proc. of ACM SIGCOMM*, Aug. 2008.

[7] B. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. Papadimitriou, and J. Kubiatowicz. Selfish caching in distributed systems: a game-theoretic approach. In *Proc. of ACM Symposium on Principles of Distributed Computing (PODC)*, July 2004.

[8] G. Dán. Cooperative caching and relaying strategies for peer-to-peer content delivery. In *International Workshop on Peer-to-peer Systems (IPTPS)*, Feb. 2008.

[9] G. Dán, T. Hossfeld, S. Oechsner, P. Chołda, R. Stankiewicz, I. Papafili, and G. Stamoulis. Interaction patterns between P2P content distribution systems and ISPs. *IEEE Commun. Mag.*, Revised Aug. 2009.

[10] S. Dykes and K. Robbins. A viability analysis of cooperative proxy caching. In *Proc. of IEEE INFOCOM*, pages 1205–1214, 2001.

[11] M. Fujita, S. amd Yamashita and T. Kameda. A study on r-configurations - a resource assignment problem on graphs. *SIAM J. Discrete Math.*, 13:227–254, 2000.

[12] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *Proc. of ACM IMC*, pages 35–48, 2005.

[13] M. Hefeeda and O. Saleh. Traffic modeling and proportional partial caching for peer-to-peer systems. *IEEE/ACM Trans. Networking*, 16(6):1447–1460, 2008.

[14] IETF Application Layer Traffic Optimization Working Group (ALTO). http://www.ietf.org.

[15] Internet Cache Protocol v2, rfc2186. http://www.ietf.org/rfc/rfc2186.txt.

[16] Ipoque. Internet Studies 2007. http://www.ipoque.com, 2007.

[17] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *Proc. of Internet Measurement Conference*, pages 63–76, 2005.

[18] S. Khot. Improved inapproximability results for MaxClique, chromatic number and approximate graph coloring. In *Proc. of IEEE Symp. on Foundations of Computer Science*, pages 600–609, Oct. 2001.

[19] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis. Distributed selfish replication. *IEEE Trans. Parallel Distrib. Syst.*, 17(12):1401–1413, 2006.

[20] S. Le Blond, A. Legout, and W. Dabbous. Pushing bittorrent locality to the limit, INRIA, Tech. Rep. 0034382, Dec. 2008.

[21] N. Leibowitz, A. Bergman, R. Ben-shaul, and A. Shavit. Are file swapping networks cacheable? Characterizing P2P traffic. In *Proc. of 7th Int. Workshop on Web Content Caching and Distribution (WCW'02)*, Aug. 2002.

[22] P. Marciniak, N. Liogkas, A. Legout, and E. Kohler. Small is not always beautiful. In *International Workshop on Peer-to-peer Systems (IPTPS)*, Feb. 2008.

[23] MultiProbe Project. http://multiprobe.ewi.tudelft.nl.

[24] J. F. Nash. Equilibrium points in n-person games. *Proc. of the Nat. Academy of Sci. (PNAS)*, 36(1):48–49, 1950.

[25] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang. In search of the elusive ground truth: the Internet's AS-level connectivity structure. In *Proc. of ACM Sigmetrics*, pages 217–228, June 2008.

[26] OverCache P2P. http://www.oversi.com.

[27] PeerCache. http://www.joltid.com/index.php/peercache.

[28] P. Rodriguez, C. Spanner, and E. Biersack. Analysis of web caching architectures: hierarchical and distributed caching. *IEEE/ACM Trans. Networking*, 9(4):404–418, 2001.

[29] G. Salaita, G. Hoflund, S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive web caching : towards a new global caching architecture. *Computer networks and ISDN systems*, 30(22-23):2169–2177, 1998.

[30] G. Shen, Y. Wang, Y. Xiong, B. Y. Zhao, and Z. Zhang. HPTP: Relieving the tension between ISPs and P2P. In *Proc. of IPTPS*, Feb. 2007.

[31] R. Tewari, M. Dahlin, H. Vin, and J. Kay. Beyond hierarchies: Design considerations for distributed caching on the Internet. In *Proc. of International Conference on Distributed Computing Systems*, pages 273–284, 1999.

[32] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Woźniak. Cache replacement policies for P2P file sharing protocols. *Euro. Trans. on Telecomms.*, 15:559–569, 2004.

[33] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. On the scale and performance of cooperative web proxy caching. 34(5):16–31, 1999.

[34] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Provider portial for P2P applications. In *Proc. of ACM SIGCOMM*, 2008.

[35] D. D. Yao. S-modular games, with queueing applications. *Queuing Systems*, 21:449–475, 1995.

Appendix

The proof we describe here follows the proof described in [24]. Before proving Theorem 1 we recall Kakutani's fixed point theorem [24].

*Lemma 2 (***Kakutani***): Let $\mathcal{B} \subseteq \mathbb{R}^{|\mathcal{H}|}$, $\mathcal{B}$ compact, convex and non-empty. Let $\mathcal{K} : \mathcal{B} \rightrightarrows \mathcal{B}$ be a correspondence (non empty valued), s.t. $\mathcal{K}(b)$ is convex $\forall b \in \mathcal{B}$. Assume, moreover, that $\mathcal{K}$ has closed reduced graph. Then, there is a fixed point for $\mathcal{K}$, i.e. $\exists b \in \mathcal{B}$ s.t. $b \in \mathcal{K}(b)$.*

The following proof of Theorem 1 consists of showing that the conditions of Lemma 2 are satisfied.

Proof: (**Theorem 1**) $\mathcal{B}_i$ is non-empty because for $K_i > 0$ there is at least one feasible relaying vector. $\mathcal{B}_i$ is closed and bounded, hence it is compact. Furthermore, $\mathcal{B}_i$ is convex due to the linearity of the cache capacity constraints (1).

The payoff function that ISP i aims to maximize is continuous in $r^h_{i,i}$ and in $r^h_{i',i}$ both for *LC* and for *NC*, and it is quasi-concave in $r^h_{i,i}$ as it is linear.

We define the set valued best response function of ISP i

$$\mathcal{K}_i(\mathbf{r}_{-i}) = \{\mathbf{r}_i \in \mathcal{B}_i | f_i(\mathbf{r}_i, \mathbf{r}_{-i}) \geq f_i(\mathbf{r}'_i, \mathbf{r}_{-i}) \text{ for all } \mathbf{r}_i \in \mathcal{B}_i\}.$$

The set $\mathcal{K}_i(\mathbf{r}_{-i})$ is non-empty because f_i is continuous and $\mathcal{B}_i$ is compact. It is convex due to the quasi-concavity of the payoff function. The graph of $\mathcal{K}_i$ is closed due to the continuity of all pay-off functions.

Let us define $\mathcal{B} = \times_{i \in I} \mathcal{B}_i$ and the correspondence $\mathcal{K} : \mathcal{B} \rightrightarrows \mathcal{B}$ as $\mathcal{K} = \times_{i \in I} \mathcal{K}_i$. $\mathcal{B}$ is hence compact, convex and non-empty, and $\mathcal{K}$ is convex, non-empty valued and has closed reduced graph. Hence, due to Kakutani's theorem $\mathcal{K}$ has a fixed point such that $\bar{\mathbf{r}}_i = \mathcal{K}(\bar{\mathbf{r}}_i)$, which proves the existence of a Nash-equilibrium both for *LC*, *NC* and for a mixture of the two strategies. ■

György Dán received the M.Sc. degree in computer engineering from the Budapest University of Technology and Economics, Hungary in 1999 and the M.Sc. degree in business administration from the Corvinus University of Budapest, Hungary in 2003. He worked as a consultant in the field of access networks, streaming media and videoconferencing 1999-2001. He received his Ph.D. in Telecommunications in 2006 from KTH, Royal Institute of Technology, Stockholm, Sweden, where he currently works as an assistant professor. He was visiting researcher at the Swedish Institute of Computer Science in 2008. His research interests include the design and analysis of distributed and peer-to-peer systems.