KTH Electrical Engineering

Resource Allocation in Operator-owned Content Delivery Systems

VALENTINO PACIFICI

Doctoral Thesis
Stockholm, Sweden 2016

TRITA-EE 2015:120
ISSN 1653-5146
ISBN 978-91-7595-835-4

KTH School of Electrical Engineering
SE-100 44 Stockholm
Sweden

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av doktorsexamen fredagen den 29 Januari 2016 klockan 10.00 i sal F3, KTH, Stockholm.

Tryck: Universitetsservice US AB

Abstract

Live and on-demand video content have become the most important source of network traffic in mobile and fixed networks in recent years. In order to be able to efficiently deliver the increasing amount of video content, network operators have started to deploy caches and operator-owned CDNs. These solutions do not only reduce the amount of transit traffic of the operators but they may also improve the customers' quality of experience, through bringing the video content closer to customers. Nevertheless, their efficiency is determined by the algorithms and protocols used to allocate resources, both in terms of storage and bandwidth. The work in this thesis proposes and analyses algorithms for the allocation of these two resources for operator-owned content management systems.

In the first part of the thesis we consider a cache maintained by a single network operator. We formulate the problem of content caching in a mobile backhaul as an integer program and we show that there exists an efficient centralized solution to the problem. Due to the prohibitive space complexity of the centralized algorithm, we propose two distributed algorithms based on local information on the content demands that compute an approximate solution. We then consider the problem of managing cache bandwidth so as to minimize the traffic cost of content delivery. We propose a model of the problem in the framework of Markov decision processes and we propose various approximations of the optimal stationary policy. We evaluate the proposed policies through extensive simulations and experiments and we show that active cache bandwidth allocation can significantly increase traffic savings.

We then consider the interaction among content management systems maintained by different network operators. First, we consider the problem of selfish replication on a graph as a model of network operators that use their caches to prefetch popular content, and try to leverage their peering agreements so as to minimize the traffic through their transit providers. We use game-theoretical tools to investigate the existence of stable and efficient allocations of content at the network operators. We design efficient distributed algorithms that compute a stable content allocation through selfish myopic updates of content allocations at different network operators. We show that, if the cost function is neighbor-specific, network operators need bilateral payments to compute a stable content allocation that is individually rational. We then consider the problem of coordinated caching in a network of autonomous systems engaged in content-level peering. We show that interacting operator-owned caches can reach a stable content allocation without explicit coordination. However, peering network operators that coordinate to avoid simultaneous cache updates converge to a stable content allocation more efficiently. Finally, we show that peering operators are likely to implement cost efficient cache allocations even if the estimate of the content popularity is inaccurate.

Beyond the theoretical contributions made to the analysis of player-specific graphical congestion games and their generalizations, the results in thesis provide guidelines for the design of protocols for standalone and for interconnected operator-owned content management systems.

Acknowledgments

I would like to thank my advisor György Dán for his guidance and for all the fruitful discussions that provided me with fundamental insights into the problems I was going to address. I sincerely appreciate his patience as well as his dedication to push me forward, even when things did not turn out as expected. I enjoyed every moment I spent discussing and working with him and I wish my next supervisor to be as sharp, sympathetic and charismatic. The bar has been set pretty high.

I would also like to thank professor Gunnar Karlsson for giving me the opportunity to become a member of LCN. His genuine interest into making LCN an efficient and enjoyable working place yielded countless benefits and makes a big difference every day.

Furthermore, I am happy to thank all the members of LCN who maintain a joyful environment in the lab and break the monotony of everyday work. Stelios, Emil, Slađana and Sakis, for being always cheerful and fun.

I am thankful to all my friends, in Stockholm and abroad, who have encouraged, entertained and supported me through these years. In particular, I would like to thank my friend Tanja, for being always so responsible, trustworthy and sensitive, and Marco, for reminding me that there is always an alternative way to go.

Last but not least, I would like to express my greatest gratitude to my loved ones; this journey would not have been possible without the constant support of my family. To my father Walter for believing in me from the start and for pushing me forward when it was most difficult. To my mother Mirella and my sister Roberta for their love.

Contents

Chapter 1

Introduction

1.1 Background

In recent years, the usage patterns of the Internet have increasingly shifted towards content generation, distribution and sharing. Real-time and on-demand video are widely consumed by Internet users and video content has become one of the most important sources of network traffic. In 2014, Netflix [1] was the leading downstream application in North America and together with YouTube accounted for nearly 50 percent of downstream traffic on fixed networks. Facebook traffic increased by over 200% on fixed networks, driven mainly by the roll out of the autoplay feature for videos [2]. Cisco predicted a three-fold increase in IP traffic by 2019 [3], more than 80 percent of which is expected to be video traffic.

Due to the increasing demand for content, efficient content management is crucial to improve the performance and to reduce the cost of content delivery. Content providers are continuously deploying new infrastructure to face the growth of user demand for content. At the same time, content providers outsource content delivery to commercial content management systems (CMSs), such as content delivery networks (CDNs). CDNs provide a scalable solution to the problem of distribution of content to Internet users and are widely adopted for the delivery of video content; more than half of all Internet video traffic crossed CDNs in 2014, and Cisco predicted that more than two-thirds of all video traffic will cross CDNs by 2019 [3]. The market size of CDN is expected to grow from $ 4.95 billion in 2015 to $ 15.73 billion in 2020 [4].

1.2 Motivation

While the business of commercial CDNs thrives, network operators must face the increasing growth of Internet traffic without being part of the revenue-distribution mechanism of the content delivery market. Increased demand for traffic puts stress on operators' infrastructure and affects the quality of experience (QoE) of the subscribers. Furthermore, dynamic content workload and CDN content placement and redirection policies introduce new challenges for network operators trying to optimize their networks.

In response to these challenges, many service providers have deployed own content management systems [5, 6, 7, 8], with the objective of controlling and optimizing the delivery of content in their own network. By serving content from within their networks, operators aim at reducing their cost for traffic and increase the quality of experience of their subscribers. In addition, they may become part of the content delivery revenue chain by distributing publishers' original content to their subscribers.

Network operators deploy internal content management systems either by setting up their own infrastructure or by involving a CDN provider [9]. In the latter case, the CDN provider can either deploy its own servers and operate the CDN on behalf of the network operator, or it can license its software to the network operator, which is in charge of deploying and operating the CDN itself. The former approach is referred to as *managed CDN* [10], while the latter approach is called *licensed CDN* [11, 12].

In either case, network operators designing and deploying their content management systems need to solve two main problems. First, they need to characterize the workloads generated by their subscribers, to efficiently dimension and organize their storage sites. Second, they have to solve problems of resource allocation for optimal content delivery.

We consider this resource allocation problem first in the context of a single network operator and we investigate whether efficient content placement can be achieved in the content management system of a single network operator. Furthermore, we investigate how content delivery at a single network operator affects the content distribution system as a whole, in terms of costs and efficiency. We then consider the potential interaction among content management systems maintained by different network operators. We investigate the effects of interaction and coordination among network operators that leverage the content allocated at the connected content management systems, so as to improve the performance of content delivery within their own networks.

1.3 Thesis Structure

The structure of this thesis is as follows. In Chapter 2, we introduce performance metrics that network operators use to evaluate their content management systems and we describe different aspects of the problem of resource allocation. In Chapter 3, we investigate the problem of resource allocation in a stand-alone content management system deployed by a single network operator. In Chapter 4, we extend our analysis by considering a network of content management systems deployed by multiple autonomous network operators and we focus on the effects of the interaction among interconnected network operators. Chapter 5 provides a summary of the papers included in this thesis along with the thesis' author contributions to each paper. Chapter 6 concludes the thesis by summarizing the main findings and discussing potential directions for future research.

Chapter 2

Resource Allocation in Content Management Systems

Storage capacity, content placement, available bandwidth and routing of user requests in a content management system are arguably the key aspects of the resource allocation problem that affect cost and efficiency of content delivery. The storage capacity and the placement strategies of content items affect the share of user requests that can be served within the operator's network. Together with the routing of user requests and the bandwidth allocated to the content management system, they affect the costs incurred by the operator and influence the users' QoE. In this chapter we describe some of the performance metrics that network operators use in order to quantify the effects of these aspects on the efficiency of content delivery. In addition, we introduce the problem of resource allocation in content management systems.

2.1 Performance Metrics

In the following we discuss two performance metrics that are useful to assess the performance of operator-managed content management systems.

Content Delivery Cost

When deploying and operating a content management system, a network operator incurs capital (CapEx) and operational (OpEx) expenditures, that, together, likely make up the most significant share of the operator's aggregate costs [13]. Therefore, it is in the operator's interest to efficiently dimension and utilize the infrastructure for content delivery.

The storage and the bandwidth allocated to the content management system influence the costs of the network operator directly, upon provisioning, and indirectly, through affecting performance. The bandwidth at which content is served affects

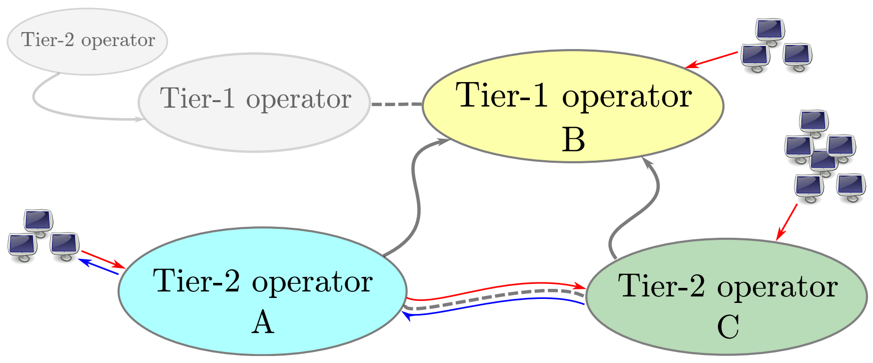


Figure 2.1: A schematics showing an example of business agreements among tier-1 and tier-2 network operators. The continuous arrows are transit links while the dashed lines are peering links. In the figure, the request from one of the subsribers of operator A is served through a peering link from operator C.

the subscribers' QoE and in turn the future profit of the operator. The amount of storage allocated to the content management system determines the share of content that can be served from within the operator's network, and thereby affects the amount of inter-operator traffic generated by content retrieval. The cost of inter-operator traffic may vary depending on the business agreements stipulated with the neighboring network operators.

Inter-operator business agreements

In today's Internet, netwok operators are commonly classified into tiers. Tier-1 network operators connect their networks together in peering relationships and do not need to purchase traffic to reach every other network on the Internet. Tier-2 and 3 network operators connect to other networks in order to deliver their subscribers' traffic to destinations outside their footprint, and typically establish client-provider business relations (transit agreements) with one or more higher tier operators, in order to ensure global connectivity. In addition, network operators typically maintain peering agreements with adjacent autonomous systems, with the purpose of mutually reducing their transit traffic. The traffic exchanged between peering operators is usually not charged, unlike the traffic that low-tier network operators exchange with their transit providers. In Figure 2.1 we provide an example of business agreemnts among tier-1 and tier-2 network operators.

The cost for transit accounts for a share of the OpEX of tier-2 and 3 network operators that is difficult to quantify, since network operators usually do not publish the details of their operational costs. Nevertheless, it is in the interest of tier-2 and 3 network operators to minimize traffic delivered through their transit providers, by leveraging their settlement-free peering agreements. This can be done through appropriate content placement strategies, efficient algorithms of user-request routing and agreements for cooperation between peering operators.

Quality of Experience

Over-the-top content providers such as Netflix[1], Hulu [14], and Amazon [15], try to improve customer satisfaction through increasing the QoE of their subscribers. As one of the main factors that influence the subscribers' QoE is the quality of the delivered content, content providers commonly offer 3D content, and super HD content has become available recently [1]. Higher quality content requires increased bit-rates which in turn stress network operators' infrastructures. In addition to the central role played by the bit-rate in the delivery of video content, several over-the-top content providers have highlighted the critical effects of perceived latency on user satisfaction and on business metrics [16, 17]. As a consequence, both access latency and bit-rate are important performance metrics for the problem of resource allocation in content management systems.

2.2 The Resource Allocation Problem

When designing and deploying content management systems, network operators strive to provide satisfactory QoE to their subscribers at reasonable costs, by allocating their resources effectively. The problem of resource allocation in a content management system models the decision making process of a network operator aiming at maximizing the performance achieved by content delivery. In the following we briefly describe three interdependent aspects of the resource allocation problem.

Storage Capacity Dimensioning

Network operators can decide the amount of storage allocated to their content management systems. Storage capacity dimensioning can be performed in the system design phase or dynamically, in response to shifting subscribers' demands or performance goals. When designing their content management systems, network operators face the trade-off of storage cost versus returns in terms of performance metrics. By increasing the storage capacity of their content management systems, network operators become able to serve a bigger share of the content requested by their subscribers from their internal networks, thereby reducing transit traffic and improving QoE. Nevertheless, storage capacity affects CapEx, for its purchasing and deployment, and OpEx, for its maintenance.

In this context, one of the main challenges faced by network operators is to correctly estimate the added value provided by each unit of storage, so as to solve the trade off of storage cost versus performance gains.

Content Placement

Network operators can optimize content delivery by selecting the content stored in their content management systems. Typically, it is possible for network operators to collect statistics of user accesses to content, with the purpose of estimating the expected future popularity of each content item among their subscribers [18, 19]. The information on the expected popularity of content items can then be used to optimize the placement of content in the storage of the content management system.

We distinguish between two processes of content placement in content management systems.

Content caching: A *caching decision* is taken upon the retrieval of a content item requested by a subscriber but not available in the storage of the content management system (i.e., cache miss). The network operator decides whether to store the content item, partially or in its entirety. The decision is taken according to the *cache admission policy* implemented in the system. If the decision is to store the content, the network operator uses a *cache eviction policy* to decide on the content that should be evicted to make room for the new content item.

Content replication: When performing *pre-fetching* or *replication*, a network operator computes the set of content items to store in its content management system. The computation is usually done by solving an optimization problem based on the expected future popularity of the content among the subscribers and using the performance metrics described in Section 2.1. The content items in the optimal set are then retrieved from their origin during periods of low demands, and are stored in the content management system. Upon changes of expected content popularity or expected costs and latencies, the network operator recomputes the optimal set of content items and updates the set of allocated content by actively fetching new content items.

The complexity of the optimization problem, the lack of new predictions on future content popularity and the size of the content items to be retrieved pose constraints on the amount of pre-fetching operations that can be performed in a give time span. For this reason, the time period between two subsequent pre-fetching operations is usually significantly longer compared to the time period between subsequent caching decisions.

In both cases of content replication and content caching, the set of content items stored in the content management system affects the efficiency of content delivery by influencing the cache hit rate and, consequently, the amount of transit traffic and the user-perceived latency.

Bandwidth Allocation

By using multiple techniques of traffic inspection [20, 21, 22], network operators can categorize user traffic and distinguish among traffic flows generated by different Internet services or subscribers. When facing constraints of bandwidth or in moments of high user activity, network operators may have an incentive to prioritize among the traffic flows, thereby guaranteeing higher bandwidth to profitable applications, like the delivery of pay-per-view content, or providing better QoE to their premium customers.

Similarly, network operators might profit by actively allocating the bandwidth of their content management systems. By influencing the rate at which each single content item is served to the subscribers, they would act on the subscribers' QoE and therefore influence the permanence and the future access patterns of users in the system. In some systems, the effects of bandwidth management on the costs incurred by network operators can be relevant [23, 24]. Therefore, when optimizing the allocation of bandwidth to content, network operators should take into account its direct and indirect effects on the performance metrics.

It is worth mentioning that active bandwidth allocation based on traffic categorization would disregard some of the principles of net neutrality [25] and its practice might be illicit [26, 27].

Chapter 3

Stand-alone Content Management Systems

A network operator designing, deploying and operating its own content management system implements algorithms and protocols with the objective of optimizing content delivery in its own network. While aiming at maximizing its profit, the operator evaluates the performance of the content delivery process by monitoring its expenditures and the QoE of its subscribers. In the following we focus on the effects on the performance metrics of a single network operator of storage capacity, content and bandwidth allocation. In this chapter, we do not consider the potential effects of the operator's actions on the content delivery ongoing at neighboring network operators.

3.1 Storage Capacity Dimensioning

Storage capacity allocation in networked systems has received limited attention in the literature. The authors in [28] proposed two approaches to determine the optimal storage capacity based on expected content popularity and on costs of memory and bandwidth, for the case of web caches. Most of the remaining work related to cache systems focuses on cache eviction policies, cache placement and routing of requests. The reason can be mostly attributed to the combination of wide availability of cheap storage and limited size of web objects, which made the problem of cache dimensioning less relevant in practice. Consequently, assumptions such as unlimited caching storage became common [29, 30, 31].

In recent years, video-on-demand (VoD) systems have become commonplace. Cisco predicted that consumer VoD traffic will double by 2019 and HD will be 70 percent of IP VoD traffic in 2019 [3]. The use of multiple replica sites close to the users for achieving efficient video content delivery is now the norm [32, 33, 34, 35], caching of P2P content is already performed by network operators [36, 37]. Unlike small sized web objects, video content and P2P shared content can easily deplete the capacity of a content management system, even under the current prices for storage. In such scenario, network operators deploying a content management system might need to characterize the necessary amount of storage required to meet the requirements of QoE for their customers.

The internal structure of the content management system often plays a central role in assessing the optimal storage capacity allocation. When dealing with multiple replica sites for example, network operators could opt for a multi-tier architecture. In such systems, the optimization of storage size would need to take into account the different performance requirements and costs for each tier. The problem of storage capacity allocation for VoD systems has been investigated in [38, 39] where the authors considered hierarchical content distribution systems and characterized the resource requirements to meet given goals on the delay perceived by the subscribers. The performance benefits of jointly optimizing content placement and the storage capacity allocation have been investigated in [40]. The authors showed that a greedy approach to the problem can achieve performance very close to the optimal.

3.2 Content Placement

Network operators can improve the performance of content delivery by placing the most popular content items close to their subscribers. It is known from results from disk and memory caching [41, 42] that optimal allocation decisions should be made so as to evict content that will not be accessed furthest ahead in the future. In practice however, the future access pattern is not known. In addition, content items often have different sizes and variable retrieval costs, which makes the problem of content placement fundamentally different from traditional caching [43].

Content Caching

If network operators lack an accurate prediction of future content popularity, they have to update the content allocation continuously, upon requests of subscribers for content items. There is a wide range of work that focuses on the design and evaluation of cache eviction policies [44, 45]. The simplest policies proposed in the literature try to capture and exploit the temporal locality of reference in user-request streams. The Least Frequently Used (LFU) and Least Recently Used (LRU) eviction policies exploit two different forms of temporal locality, which are, respectively, locality due to popularity and locality due to recency of references. LFU keeps track of the frequency of user accesses to content items and discards the least often used item, while LRU keeps track of the recency of user accesses and discards the least recently used item.

LFU is optimal under an independent reference model [41], but it is oblivious to recency of access. Furthermore, its running time per request is logarithmic in the cache size. LRU is an approximation of the optimal replacement decision whenever the recent history of user accesses is an approximation of the future. LRU is simple to implement, and tends to perform considerably well, and thus it is widely used in Web servers, client applications and proxy servers [46]. In addition, there exist simple approximate models to accurately estimate the hit probability of LRU under different user request arrival models [47, 48]. Nevertheless, both the LFU and the LRU cache eviction policies base their eviction on one single form of temporal locality of reference. In order to capture both recency and frequency of references, some hybrid policies like LRFU [49], LFU with Dynamic Aging (LFU-DA) [50], and ARC [51] have been proposed instead. Experiments showed that LFU-DA performs better than most existing algorithms in terms of hit rate [50].

The eviction policies discussed above base their decisions exclusively on temporal locality of reference, and aim at maximizing the hit rate of the cache. In case user access patterns to content present little temporal locality of reference (e.g. long-tailed content popularity distribution), requests to less popular content might result in the eviction of popular content items. This phenomenon, referred to as *cache thrashing*, might negatively affect the cache hit rate, and can be mitigated through the utilization of randomized or adaptive cache replacement policies [52].

While hit rate maximization leads to lower access latency for content, it might achieve poor performance in terms of traffic cost minimization [53, 54], as in most systems the cost for content retrieval varies among different items and depends on a variety of factors that include heterogeneous content size and traffic costs. The GreedyDual-Size (GDS) [43] eviction policy evicts content items with the smallest key value for a certain cost function. The key value is proportional to the cost of the content item which, in turn, depends on the goal the algorithm wants to achieve. GDS can be set to minimize the average latency or the overall traffic, and it has been shown to outperform most of the other replacement policies. One shortcoming of GDS is that it does not take into account how many times the content item has been accessed in the past. GDSF [50] and GDSP [55] are extension to GDS that consider also locality due to popularity.

Content Replication

In some cases, network operators may be capable of accurately predicting the future popularity of content items [51, 56, 18, 57]. The predicted content popularity would allow operators to to update the storage allocation of their content management systems by making effective pre-fetching decisions. In [58], the authors showed that pre-fetching of content based on prediction of future content popularity can achieve hit ratios comparable to conventional LRU caching, with the additional benefit of significantly lower storage capacity requirements and higher bandwidth savings.

In order to compute the optimal content allocation given the future popularity of content items, network operators need to solve a $0-1$ Knapsack problem which is known to be weakly NP-hard (i.e. a dynamic programming solution which run in pseudo-polynomial time exists). In addition, the $0-1$ Knapsack problem has a fully polynomial time approximation scheme (PTAS) [59].

Pre-fetching can be done at different time scales. A large time period between consecutive pre-fetching operations would not allow to track the variability of user access trends and thus to leverage the short term fluctuations in the content popularity. In [54] the authors investigate whether pre-fetching can be dynamically adapted to changing predicted content popularities. They consider hybrid content distribution systems that combine cloud and CDN based infrastructure, and show that dynamic allocation can provide significant gains compared to static allocation and compared to LRU caching.

Content Placement over Multiple Storage Sites

In order to further increase the performance of content delivery, network operators often distribute the storage of their content management systems over multiple sites. A distributed storage system can push content closer to the network edge and therefore closer to the subscribers, enabling a better exploitation of locality. Content placed closer to the subscribers can reduce in-network traffic and can be accessed at lower latency [60]. Furthermore, multiple storage sites distributed across the subscriber population allow to mitigate the effects of content popularity differences across heterogeneous subscribers communities. The problem of determining the location of the storage sites in the network so as to maximize performance has been explored in [61, 60], where the authors use simulations and experiments to show that a greedy approach to the placement of storage sites on a restricted set of locations yields performance close to optimal. In addition, the authors in [60] showed that intelligent strategies of storage sites placement play a central role, as over provisioning of storage sites leads to a rapid decrease of the performance returns.

When trying to leverage the positive aspects of distributed content delivery, network operators attempting to maximize the overall performance of their content management systems need to solve the problem of placement of content items at the different storage sites. In the following we discuss some results from literature in the context of content caching and replication over multiple storage sites, and we describe the contributions of this thesis.

Content caching: Systems of interconnected caching nodes have been shown to generally improve the performance of content delivery, both in terms of user perceived latency and traffic cost [30]. In such networks of interconnected caches, the arrivals of requests for content that reach internal storage sites are a result of the cache misses generated at the prior nodes in the request path. As a consequence, the request streams within the cache network present different characteristics compared to request streams at single caches, and traditional assumptions for stand-alone caching systems, such as the Independent Reference Model (IRM) for request streams, are violated [62].

Analytical frameworks for evaluating the performance of interconnected caches as well as for characterizing the cache system's steady-state dynamics have been proposed in [63] for a hierarchy of LRU caches and in [62, 64, 65, 66], for arbitrary topologies of caches and a wider range of cache replacement policies. In particular, the authors in [64] analyzed the impact of the initial system state on the selection of the system's steady-state, and the authors in [65, 66] proposed analytical frameworks to estimate the cache hit probability, assuming stationary and dynamic content popularity, respectively.

Due to the distributed nature of cache networks, traditional caching policies (e.g. LRU) might lead to inefficient content allocations, as they generally store each fetched content item at all intermediate caches along the item's request path. Caching policies that limit the number of copies stored on the reverse path upon fetching have been shown to make a more efficient use of the caching storage, and to significantly improve the hit rate of the caching system [47, 67].

The works above evaluate the caching policies with respect to the achieved hit rate, and they neglect the traffic cost for content retrieval. [53] shows that maximizing the hit rate could lead to cache allocations that are suboptimal in terms of the cost for traffic incurred by the network operator.

Content replication: In the context of replication, one important question is whether a network operator can implement an optimal placement of content to caches, given the characteristics of the distributed content management system and the popularity of content among subscribers. As the set of content items available to the subscribers is very large, computing an optimal content placement might be prohibitive, in terms of both space and computational complexity, as it would entail to solve a $0-1$ integer linear program. The problem of finding an optimal content placement in arbitrary network structures has been investigated in [68, 69] and it has been shown to be NP-hard. The authors in [68, 69, 70, 71, 72] proposed computationally efficient algorithms that lead the system to a reasonable approximation of the optimal solution. In particular, [72] proposed a set of centralized, polynomial time algorithms with approximation guarantees, for the joint problem of request routing and content replication under strict bandwidth constraints at the storage sites. Such algorithms are centralized and based on the assumption of full information about content popularities at the storage nodes. As in large systems the full information about the popularity of content at each network node might be difficult to collect in a central node, two important questions are whether there exist low complexity, distributed algorithms that optimize content placement, and whether an efficient content placement can be computed using the limited information available at each storage site.

Game theoretic analyses of distributed selfish replication on graphs can serve as a basis for lightweight distributed content placement algorithms. The learning dynamics typically provide decentralized rules that lead the system to a Nash equilibrium [73, 74, 75, 76]. Results on the approximation ratio of such learning dynamics, referred to as price of anarchy (PoA), were provided in [29, 77, 78]. [29] considered caches with storage cost and unlimited storage capacity, and showed that the PoA can be unbounded if no assumptions are made on the topology of the peering graph. The authors in [77] considered capacitated caches and homogeneous costs on a complete peering graph. They showed that the PoA can be unbounded since, in general, it is directly proportional to the largest unit cost to retrieve content items.

As the PoA is unbounded in general, distributed algorithms that compute Nash equilibria fail to guarantee a bound on the cost of the approximation they achieve, in the most general case. However, bounds on the approximation ratio might emerge if some assumptions are made on the structure of the network connecting the storage sites and the popularity of content at the different storage sites. [78] proposed a selfish cooperative distributed algorithm that achieves performance close to the optimal, when the content popularity is similar at different storage sites. Similarly, efficient content placement strategies may arise for specific network topologies. [79, 80, 81] explored the problem of content placement in hierarchical networks. In particular, it has been shown in [79] that the optimal content allocation in two level hierarchical networks has a regular structure and there exists a greedy distributed algorithm that achieves a 2-approximation, as the set of feasible content allocations is a matroid and the objective function is submodular [82]. The authors in [80] provided an algorithm for computing the optimal placement in a hierarchical network where the storage sites are placed at the leaves of the hierarchy, by reducing the content placement problem to a minimum-cost flow problem. Motivated by the computational complexity of the problem, they design, and further improve in [81] a distributed amortizing algorithm that achieves a constant factor approximation. The model considered in [80, 81] is based on the ultrametric cost model introduced in [83].

While ultrametrics have been widely used to model hierarchical networks, they exclude the presence of storage sites at internal nodes of the network. In addition, ultrametrics are characterized by the the assumption of symmetric costs between storage sites, which is, in some cases, limiting. The bandwidth demands on links connecting storage sites might present variable characteristics of cost and latency, as most of the traffic is flowing down-link, especially during peak-time [84]. In such scenarios, efficient distributed algorithms could make use of the idle upload capacity during busy hours. Whether network operators can implement efficient content placement strategies on hierarchical network topologies through low-complexity distributed algorithms is still an open question. In this thesis we contribute to answer this question.

In Paper A, we formulate the problem of content caching in a mobile backhaul during peak hours and we propose two low-complexity distributed algorithms based on limited information on the content popularities that can be used to solve large problem instances. We provide analytical and numerical results on the approximation ratio and on the efficiency of the algorithms in computing a content allocation. We show that distributed algorithms based exclusively on local information available at the single storage sites might be not be able to achieve good cooperative caching performance. Nevertheless, we show that the proposed algorithm that requires limited levels of information about the surrounding storage sites achieves consistently a good performance, even compared with a greedy optimization based on global information which achieves a 2-approximation.

3.3 Bandwidth Allocation

Bandwidth management is one of the tools that network operators can use to tackle the increasing traffic demand that, in turn, affects the cost of traffic and threatens the QoE of subscribers. By actively engineering the user traffic, network operators can guarantee throughput for the most important services or meet the QoE requirement of premium subscribers, even in case of high network load. Traffic engineering has been shown to provide significant benefits to network operators [85], and commercial solutions that allow categorization of traffic using deep packet inspection (DPI) are available [20, 21, 22].

However, the emergence of new Internet applications and services constitutes a challenge for network operators trying to categorize user traffic. Furthermore, penalizing the most bandwidth and latency demanding applications like video-on-demand (VoD), peer-to-peer (P2P) and voice-over-IP (VOIP), degrades the QoE of the subscribers using these services that could potentially leave the network operator [86], and violates the principles of net neutrality [25]. Despite this, today it is common among network operators to block or throttle VOIP and P2P traffic [87]. As an alternative approach, some network operators have deployed P2P caches to address the problem of increased traffic demands from P2P [36, 37]. P2P caches decrease the amount of inter-ISP traffic by storing the most popular content in the operator's own network, saving the cost of content retrieval from external networks.

While bandwidth management of user traffic has been extensively investigated [85, 88, 89], the impact of the cache bandwidth and its management has received little attention in the literature. This is partially due to the fact that bandwidth management is not necessary in Web caching, as the amount of data served from the cache equals the traffic savings achieved by caching. Nonetheless, bandwidth management affects the subscribers' QoE and in turn influences their churn in the system. While this effect is negligible for Web caching, in the case of P2P-like systems bandwidth allocation affects the characteristics of the overlay, and the short and long term impact on the traffic cost may be relevant [23, 24].

Therefore, one important question is whether a network operator should allocate cache bandwidth among competing P2P overlays. Several works propose schemes for bandwidth allocation among multiple P2P overlays [90, 91, 92], but their purpose is to maximize the total download rate of the system in an attempt to minimize the download latencies perceived by the peers.

In this thesis we contribute to answer the question of whether cache bandwidth allocation could be used by network operators to minimize their traffic costs. In paper B, we model the decision of several network operators that perform active cache bandwidth allocation. We prove the existence of an optimal decision policy that only depends on the current state of the system. We propose various P2P cache bandwidth allocation policies that approximate the optimal policy and we demonstrate the importance of capturing the impact of the cache on the characteristics of the P2P overlays. By performing simulations and experiments, we show that P2P cache bandwidth should be actively managed in order to achieve significant gains in terms of traffic cost, and we identify some of the primary factors that influence the gains.

Chapter 4

Network of Operator-owned Content Management Systems

Recent industry efforts aim at interconnecting the content management systems deployed by different network operators [93, 94]. Such interconnection could benefit network operators and content providers alike. Network operators would be able to leverage their business agreements with connected providers in order to decrease their transit traffic and to optimize content placement and request routing so as to decrease the average latency perceived by their users. Along with network providers, content providers would benefit from a transparent solution for bringing content closer to their customers. The success of networks of operator-owned content management systems depends on the one hand on developing appropriate interfaces and signaling systems, and on the other hand on the design of algorithms and protocols that perform well with respect to the performance metrics discussed in Section 2.1.

In this chapter we describe three cases where the interconnection of operator-owned content management systems might be beneficial for network operators. For each of the cases we discuss, we investigate the effects of the interaction among the autonomous content management systems on the cost incurred by the network operators and on the QoE perceived by the operators' subscribers.

4.1 Content Delivery Networks

As over-the-top content providers maintain customer loyalty and increase sales by providing better QoE, the traffic generated by digital video content delivery is anticipated to show tremendous growth over the coming years [3]. To face the new challenges brought by increasing demands for digital content and in order to maintain a competitive QoE for their users, it is often more convenient for the content providers to outsource content delivery to commercial CDNs.

Through multiple replicas and efficient routing of users to replicas [95], commercial CDNs provide better performance than a system based on a single delivery server [96, 97]. In addition, they provide dynamically scaling bandwidth to cope with the variability of customer demands. Overall, they offer a relatively low delivery cost compared to the costs incurred by the content providers that deploy their own infrastructure.

Outsourcing content delivery to a single CDN does not usually provide content providers with enough footprint to serve all their customers. As a consequence, users that are not included in the CDN footprint may experience a degraded QoE. In order to overcome such shortcomings, content providers stipulate agreements with multiple CDNs, in an attempt to combine their footprints and thereby reach all of their customers with a satisfying QoE. However, in the process of choosing multiple content delivery networks, content providers have to face the complexity of negotiating business and technical arrangements with multiple parties. Once establishing the deals, content providers need to deploy ad-hoc systems to perform analytic, reporting and control over the delivery of content to the users in the footprints of the various CDNs.

For this reason, several content providers together with a few commercial CDNs have expressed the need for mechanisms for CDN brokerage. A *CDN broker* [98] or *primeCDN* [94, 99] is an intermediary between the content provider and multiple commercial CDNs. A CDN broker would be in charge of aggregating the offers of several *sub-CDNs*, i.e. wholesale content delivery networks not involved in the negotiations with the content providers. The function of the CDN broker would be to select the sub-CDNs so as to provide cost-effective delivery and optimal geographical coverage of the content providers' customers. The CDN broker would resell the offers of the sub-CDNs as a packaged service to the content providers, thereby offering hosting and delivery services, together with analytical tools to monitor the process of content delivery. In Figure 4.1 we provide an example of business agreements between content providers, CDN brokers and sub-CDNs, with arrows showing the direction of the money flows.

As today's commercial CDNs are closed systems, with proprietary interfaces and protocols, content providers outsourcing content delivery might face daunting challenges to adapt the interfaces of the different CDNs to work under the same framework, even in presence of an aggregating service such as a CDN broker. CDN interconnection would represent a solution to the shortcomings described above, and in addition, it would bring several advantages to other players in the content delivery market. Users would be able to access a wider range of content with CDN level performance and CDNs would benefit from an extended footprint and improve their resilience to flash crowd effects.

The success of CDN interconnection depends, on the one hand, on developing appropriate interfaces and protocols, and on the other hand on investigating the effects of such interconnection on content placement and request routing.
The problem of designing interfaces and mechanisms for CDN interconnection have received significant attention. A standard for CDN interconnection is under devel-

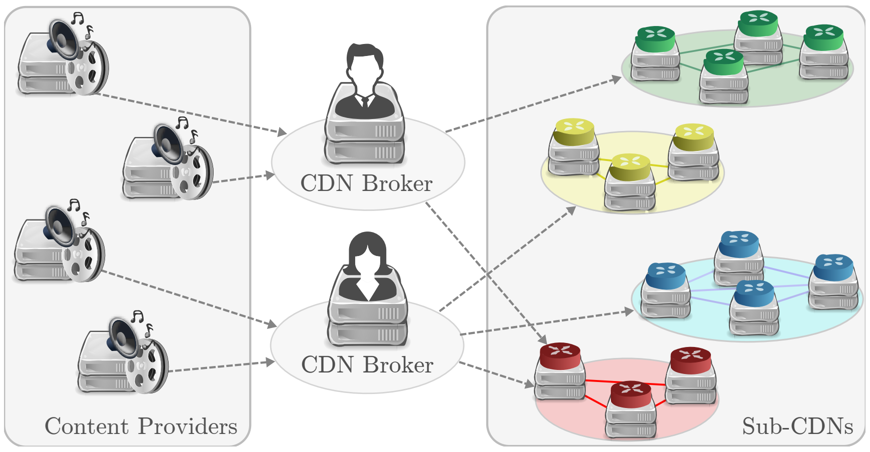


Figure 4.1: A schematics showing an example of business agreements among content providers, CDN brokers and sub-CDNs. The arrows show the direction of the money flows.

opment in the IETF CDNI working group [93]. The working group is addressing multiple issues, including the development of a framework for interconnection [100] and the design of mechanisms for request routing and redirection [101, 102], access control [103] and content naming [104].

Interconnected Operator-owned CDNs

While major over-the-top content providers try to maintain user satisfaction through increasing QoE, network operators have to cope with increased bit-rates that stress operators' infrastructures. In addition, in the traditional CDN based content distribution models, network operators are not part of the revenue chain. Many network operators have started to deploy their own *network CDNs* (nCDNs) (often referred as *Telco CDNs*, or *Carrier CDNs*) so as to cope with increasing bit-rate demands for content delivery. Recent industry efforts aim to interconnect the nCDNs through the interfaces and protocols designed in [93, 100]. By doing so, the operators attempt to improve nCDN availability and customers' QoE, and to leverage their peering agreements. Once nCDN interconnection is achieved, each network operator would then attempt to serve the content requests from its subscribers so as to minimize its traffic costs or to maximize the subscribers' QoE. An example of request routing in a network of operator-deployed CDNs trying to minimize their traffic cost is shown in Figure 4.2.

Network CDNs could pre-fetch content based on predicted demands, during periods of low demands. As network operators would be able to access the content items allocated at the connected nCDNs, successful nCDN interconnection requires

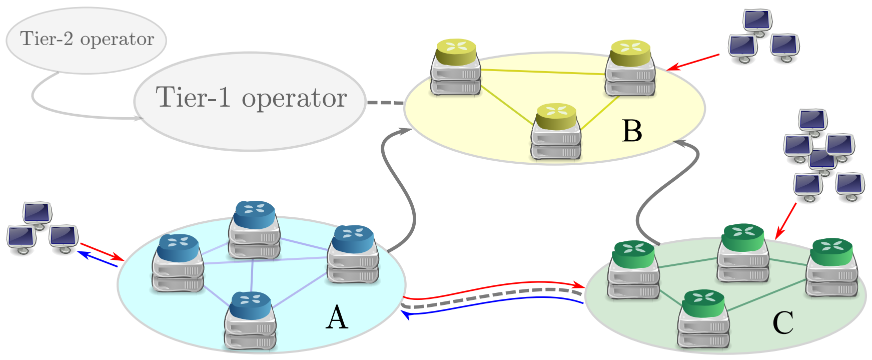


Figure 4.2: An example of routing of requests in a network of operator-deployed CDNs. The dashed lines are peering links while the continuous arrows show transit links. In the figure, the request from one of the subscribers of operator A is served from the CDN owned by operator C, through a peering link.

the network operators to be able to agree on a content allocation that, given the predicted content demands, serves all operators' interests. In lack of a central authority capable of enforcing the optimal content allocation, autonomous nCDNs would not implement the optimal solution if they can individually fare better from deviating from it. Consequently, the interaction of such autonomous, profit-maximizing entities would lead to a competitive market, where network operators follow their selfish interests.

In such a scenario, a content allocation from which no network operator can deviate unilaterally, given the content allocated at the connected nCDNs, would be compliant with every operator's interests, as long as the interconnecting topology and the content popularity estimates do not change. Such content allocation would then be self-enforcing, and it corresponds to a pure strategy Nash equilibrium [105]. Two important questions are whether such a self-enforcing content allocation exists and whether the optimizations of every autonomous nCDN would lead the systems to such a self-enforcing content allocation. In the following we distinguish the two cases of operators minimizing their traffic costs and operators minimizing the latency perceived by their users.

Traffic Cost Minimization

If network operators implement nCDN interconnection, they would leverage their peering links by accessing the content allocated at the connected peering nCDNs, so as to decrease their cost for transit traffic. Additionally, they would tailor their nCDNs' content allocation in response to the decisions on content allocations made by their peers.

The problem of distributed replication on a graph, where nodes store content accessed by local or remote users, is a model of the content placement decisions taken by interconnected nCDNs. The existence of a stable allocation in distributed selfish replication was shown in the case of unit storage capacity [106] and infinite number of content items [29], in the case of complete peering graph and homogeneous cost functions [107, 108], and in the case nodes can replicate a fraction of content items [109]. [110, 111] consider the case when the cost functions are linear; [110] shows that cycles in sequential updates can exist in the case of directed peering graph.

Previous works have not addressed whether network operators would reach a stable allocation in the most general case of selfish distributed replication with operator-specific, non-linear, cost functions on an arbitrary peering graph topology. In Paper C we show that a stable allocation of content items to network operators always exists and we provide an efficient distributed algorithm to compute it. In the case of non-complete peering graphs, we show that nCDNs might cycle if they perform selfish updates of their content allocation, but they would reach a stable allocation if the updates are done in random order. Finally, we consider a cost model in which peering network operators always have incentives to allocate disjoint sets of content items. We show that in such a model there are no cycles, even if the updates are performed simultaneously by nCDNs that belong to an independent set of the peering graph.

Latency Minimization

By accessing content available at connected nCDNs, network operators can increase the content footprint of their content management systems and, possibly, decrease the average access latency to content items. While it is reasonable to assume that each operator incurs constant unit traffic cost when retrieving content stored at peering operators, content items stored at different locations are in general accessed at different latencies. Therefore, the assumption of homogeneous neighboring costs made in Paper C is rather limiting if the network operators attempt to maximize their users' QoE.

In Paper D we relax the assumption of homogeneous neighboring costs and we answer the question of whether network operators would be able to agree upon and compute a self-enforcing content allocation for the scenario when their objective is to minimize the users' perceived latency. We show that a self-enforcing allocation might not exist if bilateral payments between operators are not allowed. We design two distributed algorithms that use bilateral payments to compute a stable allocation. The two algorithms, which differ in terms of the amount of information exchanged between nCDNs, are guaranteed to converge in a finite number of iterations. Furthermore, we propose a scheme that ensures that participation according to the proposed algorithms is ex-post individually rational for all CDNs. Our simulations, performed on a real AS-level network topology and synthetic topologies, show that the algorithms have geometric convergence, and that if nCDNs reveal more private information about their content demands, the algorithms converge faster.

4.2 Content-centric Networks

In order to overcome the structural limitations of the host-to-host paradigm of the current Internet, several works have proposed a clean-slate design of the Internet architecture, where content delivery is the key function. Content-centric networking is a network model oriented to content, in which content items are addressable and the basic form of interaction is host-to-content [112, 113, 114].

In a content-centric network (CCN) caches are part of the protocol stack. Content items are stored in caches spread throughout the network, and routing is content-aware. When a user asks for a content item in a CCN, the request is routed through a set of caches in the network. If the content is found in one of these caches, then it is retrieved and served, otherwise the request reaches one of the content custodians from where it is ultimately served. Content custodians store content items permanently; they are content producers or servers.

Such an architecture has several advantages over a host-to-host paradigm. Due to the availability of storage sites throughout the network, the most popular content can be efficiently pushed close to the network's edge. Furthermore, content is intrinsically redundant in the network, and it can be placed to keep the overall network traffic minimal. In addition, caching of content items is not performed at the application layer like in today's Internet, but at the network layer. This approach drastically benefits the speed of the cache lookups and the content retrieval in case of cache hit.

Nevertheless, content-centric networking comes with a wide range of challenges that need to be addressed. As the design of CCNs lies outside the scope of this thesis, we limit ourselves to a brief overview of the most interesting problems and we point to the related research work.

As the users of a CCN are oblivious to the content location, routing of user requests is performed by content name. Therefore, the naming mechanism adopted in a CCN influences the design of the routing scheme and thereby the efficiency of content delivery. In order to avoid excessive growth of routing tables, content naming should be scalable and allow aggregation. Furthermore, content naming should be persistent, as user requests for the same name at different points in time are expected to retrieve the same content item. As the location of content items is not unique nor fixed and the ownership of content might change, it is not trivial to provide naming persistence and aggregation capabilities. Several research works have addressed the problems of content naming [112, 113] and name-based routing [115, 116, 117]. In particular, [117] proposed routing algorithms that compute the paths of user requests to content based on the content popularity estimates and on the content placement scheme implemented in the CCN.

The design of efficient content placement schemes for CCNs is also challenging. The mechanism of automatic caching supported in CCNs [112] inherently presents problems of caching redundancy. Moreover, due to the mesh topology of caches in a CCN, existing cache eviction policies are not guaranteed to achieve good performance. The design and evaluation of efficient caching algorithms for CCNs has been addressed in [62, 118, 119, 120]. The cache hit rate was investigated for cache hierarchies [118] and for general topologies [119, 62, 121]. In [119], the authors considered various network topology-aware policies to improve the overall cache hit rate in a network of caches. [121] used probabilistic caching to increase the cache hit rate and the fairness among content flows in a network of caches. Traffic cost minimization was investigated in [120], where the authors proposed four on-line content replacement schemes that aim at decreasing the operator's traffic, at the price of different levels of communication overhead among the caches. Common to these work is that they considered a single entity optimizing the placement of content in CCNs.

Autonomous Networks of Caches in a CCN

Similar to the structure of today's Internet, a future content-centric network is likely to be a network of autonomous operators, each of them maintaining and managing the infrastructure within its network. Each network operator's primary concern would be its profit and the QoE of its subscribers. Therefore, the routing and the content allocation within each network would be optimized for local performance. Each network operator would have incentives to design the routing mechanism for its subscribers' requests to reach primarily the content replicas cached within its network. In addition, instead of forwarding the requests that cannot be served locally to a transit provider, the operator would try to leverage the content cached within its peering networks. In addition, in a cache network of autonomous operators, each operator would likely implement content placement schemes that attempt to minimize its traffic costs. Consequently, each caching decision at a network operator would depend on the content cached at the operator's peering networks.

An open interesting question is whether the interaction between autonomous, profit maximizing peering networks of caches in a CCN would allow the emergence of a stable allocation of content at the caches, or, on the contrary, would lead to instability and oscillations of content allocations. In Paper E we model the interaction and the coordination between caches managed by peering network operators. We prove that irrespective of whether the network operators synchronize their cache updates, the cache allocation will reach a stable allocation if neighboring operators periodically inform each other about their content allocations. Furthermore, we show that, in order to achieve fast convergence to a stable allocation, network operators establishing peering agreements should not simultaneously update the set of cached content items. In addition, we investigate the effect on the system behavior of noise in the content popularity estimates.

4.3 Cooperative Caching of Network Operators

In order to cope with the increasing requirements for content from Internet users, network operators seek for solutions that would decrease the content delivery costs without deteriorating the users' QoE. Whether and how content management systems deployed by different operators could cooperate to serve user requests has been the object of several works.

In the context of Web caching, several signaling protocols for inter-cache cooperation have been proposed [122, 123, 124]. The Internet Cache Protocol (ICP) [122] supports discovery and delivery of content stored in nearby caches. In [123, 124] instead, each cache maintains a potentially inaccurate summary of the content available at its neighbors.

In addition to the design of discovery or dissemination schemes, cooperative caching has to be analyzed to assess the potential performance gains and benefits it provides to each cooperating network operator. Cooperative web caching has received a lot of research attention, and it has been shown that the gain from cooperation is only marginal and achieved under specific conditions [125, 126]. The authors of [126] show that the factors that mostly limit the gain from cooperation are the high dynamism of web objects and the high overhead compared to the small object size. Furthermore, the ubiquitous local caches at each client significantly reduce the hit rates in cooperative web caching.

Since in the case of content delivery for P2P and VoD systems content items are mostly immutable and significantly larger than in the case of web caching, cooperative caching may be more beneficial. Nevertheless, the case of cooperative caching in P2P and VoD systems has received little attention in the literature. Cooperative P2P caching was shown to lead to considerable improvements of the cache efficiency at the cost of a negligible overhead [127, 128]. Cooperative caching for VoD systems in the context of CCNs was shown to provide significant traffic savings at the cost of an acceptable overhead [129].

An interesting question is whether cooperative caching would provide performance benefits even when the cooperating caches are managed by multiple, autonomous network operators. Potentially, the caching decisions of interconnected network operators might lead the system to instability or to inefficient allocations of content in caches, as discussed in Section 4.2. [109] considered competing, profit-maximizing network operators deploying P2P caches and engaging in cooperative caching, showed the existence of self-enforcing content allocations and proved that the performance improvements of cooperative caching are potentially significant.

Two open questions are whether the selfish caching decisions by the network operators would lead the system to such self-enforcing content allocations, and whether network operators need coordination mechanisms governing their cache updates to achieve efficient cooperative caching. We answer these questions in Paper E, where we develop a model of the interaction of cooperating caches managed by autonomous network operators. The contributions of Paper E are discussed in Section 4.2.

Chapter 5

Summary of Original Work

Paper A: Distributed Algorithms for Content Caching in Mobile Backhaul Networks

Valentino Pacifici and Slađana Jošilo and György Dán

submitted to IFIP Networking 2016 (NETWORKING), 2016.

Summary: In this paper we consider mobile network operators that attempt to optimize content delivery in their content management systems through in-network caching of popular content during times of peak demand. We formulate the problem of content placement in a mobile backhaul as a binary integer programming problem, and we proposed a distributed 2-approximation algorithm for the problem. The 2-approximation requires full information about the network topology and the link costs, as well as about the content demands at the different caches, we thus propose two distributed algorithms that are based on local information on the content demands. We show that the distributed algorithms terminate in a finite number of steps, and we provide analytical results on their approximation ratios. We use simulations to evaluate the proposed algorithms in terms of the achieved approximation ratio and computational complexity on realistic mobile backhaul topologies.

Contribution: The author of this thesis developed the analytical model in collaboration with the third author of the article. The design of the distributed algorithms was carried out in collaboration with the second author of the paper. The second author of the paper proved that the *DFG* algorithm is a 2-approximation and that the *LGS* algorithm terminates in a finite number of iterations. The remainder of the analytical results were proved by the author of this thesis. The implementation of the simulations and the analysis of the resulting data were carried out in collaboration with the second author of the paper. The article was written in collaboration with the second and third authors.

Paper B: Cache Bandwidth Allocation for P2P File Sharing Systems to Minimize Inter-ISP Traffic

Valentino Pacifici and Frank Lehrieder and György Dán

in IEEE/ACM Transactions on Networking (ToN), November 2014.

The original version of the paper appeared in Proc. of IEEE INFOCOM, 2012.

Summary: In this paper we investigate the problem of bandwidth allocation in ISP-deployed P2P caches. We consider ISPs that actively allocate cache bandwidth among the overlays in an attempt to maximize the inter-ISPs traffic savings of the cache. We formulate the P2P cache bandwidth allocation problem as a Markov decision process and we show the existence of an optimal stationary allocation policy. We show that the complexity of finding the optimal policy is prohibitive even for a moderate number of ISPs and we propose two approximations to the optimal policy. We perform simulations and experiments and show that cache bandwidth allocation can lead to significant savings in inter-ISP traffic. Based on the insights obtained during the numerical evaluation of the approximate policies, we propose a simple, priority-based, active allocation policy that performed well both in simulations and in experiments with BitTorrent clients on Planet-lab.

Contribution: The author of this thesis developed the Planet-lab framework to run the BitTorrent overlays, implemented a distributed on-line traffic estimator of inter-ISP traffic and performed the experiment-based performance evaluation of the cache bandwidth allocation policies. The second author of the paper implemented and carried out the simulations, and analyzed the resulting data. The third author developed the analytical model, proved the analytical results and designed the cache bandwidth allocation policies. The article was written in collaboration with the third author.

Paper C: Convergence in Player-Specific Graphical Resource Allocation Games

Valentino Pacifici and György Dán

in IEEE Journal on Selected Areas in Communications, December 2012.

Summary: Resource allocation games are a model of the problem of content placement in a network of operator-owned content management systems. In this paper we consider resource allocation games played over a graph: the payoff that an allocated resource yields to a player is amortized if any of its neighbors allocates the same resource. We address the question whether in resource allocation games there exist equilibrium allocations of resources, from which no player has incentive to deviate unilaterally. We prove that Nash equilibria always exist for arbitrary graph topologies, and we compute a bound on the complexity of computing an equilibrium. We then consider the problem of reaching an equilibrium when the players update their allocation to maximize their payoffs, one at a time. We show that, for complete graph topologies, the players reach an equilibrium in a finite number of asynchronous myopic updates. For non-complete graph topologies however, the asynchronous myopic updates of the players might lead to cycles. Nevertheless, we show that if the updates are performed in random order the players would reach a Nash equilibrium. We then consider the case when the players have receive no payoff from allocating resources allocated by their neighbors and we show that there are no cycles. Finally, we relax the requirements of asynchronous updates and we propose an efficient algorithm to reach an equilibrium over arbitrary graph topology and we illustrate its performance through simulations.

Contribution: The author of this thesis developed the analytical model in collaboration with the second author of the paper, proved the analytical results concerning the most general payoff function, carried out the simulations and analyzed the resulting data. The article was written in collaboration with the second author.

Errata: Proposition 6 should read "A cycle under plesiochronous best replies might exist in resource allocation games". The proof of the proposition follows from the existence of the cycle in best replies described in Section 3.2, for a resource allocation game played over the influence graph shown in Figure 1. The order in which the players make updates in the cycling asynchronous best reply dynamic is $3, 1, 4, 2, 1, 3, 2, 4$. Since $\{1, 3\}$ and $\{2, 4\}$ are independent sets of the influence graph in Figure 1, the same cycle can occur under plesiochronous best replies.

Paper D: Distributed Algorithms for Content Allocation in Interconnected Content Distribution Networks

Valentino Pacifici and György Dán

in Proc. of IEEE International Conference on Computer Communications (INFOCOM), 2015.

Summary: In this paper we consider the problem of using distributed algorithms for computing content allocations for interconnected, operator-owned, nCDNs. We consider network operators aiming at improving the quality of experience of their customers through decreasing the average latency to access content items. We show that if every operator aims at minimizing the latency perceived by its users and bilateral payments are not allowed, then it may be impossible to compute a content allocation. For the case of bilateral payments we propose two distributed algorithms, the aggregate value compensation (AC) and the object value compensation (OC) algorithms, that converge to a content allocation that is individually rational. The two algorithms differ in terms of the level of parallelism they allow and in terms of the amount of information exchanged between nCDNs. Simulations performed on a real AS-level network topology and synthetic topologies show that the algorithms have geometric rate of convergence, and that if network operators reveal more private information about their users' content demands, the algorithms converge faster. Our results also show that the algorithms scale well with the graphs' density and the nCDN capacity.

Contribution: The author of this thesis proved the analytical results, implemented and carried out the simulations and analyzed the resulting data. The author of this thesis developed the model and wrote the article in collaboration with the second author of the paper.

Paper E: Coordinated Selfish Distributed Caching for Peering Content-centric Networks

Valentino Pacifici and György Dán

submitted to IEEE/ACM Transactions on Networking (ToN), first revision.

The original version of the paper appeared in Proc. of IEEE INFOCOM, 2013.

Summary: In this paper we consider multiple autonomous systems in a content-centric network that maintain peering agreements with each other. In order to leverage each other's cache contents to decrease their transit traffic costs, the ASes engage in content-level peering. We develop a model of the interaction and the coordination between the caches managed by peering ASes. We use the model to investigate whether ASes need to coordinate in order to achieve stable and efficient cache allocations. We show that coordination is not necessary in order to reach stable cache allocations. Nevertheless, avoiding simultaneous cache evictions by peering ASes leads to fast and more efficient convergence to a stable allocation. Furthermore, we show that it is possible to obtain insights into the structure of the most likely cache allocations for the case when the estimates of the content popularities available to the ASes are inaccurate.

Contribution: The author of this thesis developed the model in collaboration with the second author of the paper, proved the analytical results for the case of perfect information and for the general case of imperfect information, implemented and carried out the simulations and analyzed the resulting data. The proofs for the free peering case under imperfect information were carried out in collaboration with the second author of the paper. The article was written in collaboration with the second author.

Publications not included in the thesis

1. Emil Eriksson, Valentino Pacifici, and György Dán. “Efficient Distribution of Visual Processing Tasks in Multi-camera Visual Sensor Networks”. In: *Workshop on Distributed and Cooperative Visual Representation and Analysis (DCVRA).* 2015, pp. 1–6

2. Valentino Pacifici and György Dan. “Stable Content-peering of Autonomous Systems in a Content-centric Network”. In: *Proc. of Swedish National Computer Networking Workshop (SNCNW).* 2013, pp. 1–4

3. Valentino Pacifici and György Dán. “Content-peering Dynamics of Autonomous Caches in a Content-centric Network”. In: *Proc. of IEEE INFOCOM.* 2013, pp. 1079–1087

4. Valentino Pacifici, Frank Lehrieder, and György Dán. “Cache Capacity Allocation for BitTorrent-Like Systems to Minimize Inter-ISP Traffic”. In: *Proc. of IEEE INFOCOM.* 2012, pp. 1512–1520

5. Valentino Pacifici and György Dan. “A Game Theoretic Analysis of Selfish Content Replication on Graphs”. In: *Poster presented at IEEE INFOCOM* (2012)

6. Valentino Pacifici and György Dán. “Selfish Content Replication on Graphs”. In: *Proc. of the 23rd International Teletraffic Congress.* 2011, pp. 119–126

7. Valentino Pacifici, Frank Lehrieder, and György Dan. “On the Benefits of P2P Cache Capacity Allocation”. In: *Poster presented at the International Teletraffic Congress (ITC)* (2011), pp. 312–313

8. Valentino Pacifici and György Dan. “A Game Theoretic Analysis of Selfish Content Replication on Graphs”. In: *Proc. of Swedish National Computer Networking Workshop (SNCNW).* 2011, pp. 1–4

Chapter 6

Conclusions and Future Work

In this thesis, we considered network operators that deploy and operate content management systems in order to improve the efficiency of content delivery within their networks. We proposed different algorithms for content delivery and we evaluated their performance in terms of cost for traffic and QoE of operators' subscribers. In the following, we summarize the main contributions of this thesis and we outline some possible directions for future work.

In the first part of this thesis, we addressed the problem of resource allocation in a stand-alone content management system. We considered a network operator that attempts to optimize the placement of content within the mobile backhaul, so as to reduce the bandwidth cost to serve users' requests during peak hours. We addressed the question of whether the network operator would be able to compute an efficient content allocation in a distributed manner, under looser requirements of computational and space complexity compared to existing centralized algorithms.

Subsequently, we considered a network operator that deploys a P2P cache in an attempt of reducing the inter-operator traffic and we investigated the impact of active cache bandwidth allocation on the inter-operator traffic savings of the cache. We demonstrated the importance of capturing the cache bandwidth impact on the characteristics of the P2P overlays and we showed that the savings yielded by cache bandwidth allocation can be significant.

In the second part of the thesis, we considered a network of operator-owned content management systems, and analyzed the effect of the interaction of the content placement decisions at different operators. We considered interconnected network CDNs, deployed by network operators in order to minimize the operators' traffic cost and maximize the QoE of the subscribers. We investigated whether network operators would be able to agree upon a content allocation of items at the nCDNs that serves all operators' interests. We addressed the question of whether network operators aiming at minimizing their traffic cost would reach such self-enforcing allocation, if they selfishly and myopically update their placement decisions. Furthermore, we investigate whether network operators aiming at maximizing the QoE of their subscribers would be able to compute of a self-enforcing allocation with low complexity and limited information availability about competing nCDNs.

Finally, we considered a Content-centric network of operators that manage their network of caches. We addressed the question of whether the interaction between autonomous peering cache networks would lead to unforeseen instability of content allocation. We proposed a model of the interaction and the coordination between autonomous peering caches and we investigated whether peering network operators need to coordinate in order to reach a stable global content allocation of items to caches.

Future Work

There are many open problems in the area of resource allocation in content management systems. For instance, in the context of content allocation in stand-alone systems, it is an open question whether distributed algorithms with bounded approximation ratio and low computational and space complexity exist for optimizing content distribution during peak-time. In order to provide bounds on the approximation ratio of such algorithms, it might be necessary to introduce assumptions on the structure of the content popularity.

In the context of interconnected, operator-owned content management systems, the effects of the interaction between interconnected, autonomous content management systems needs to be further investigated. In this thesis, we showed that self-enforcing content allocations that meet every network operator's interest might not exist if bilateral payments between network operators are not allowed. In case network operators cooperate by exchanging resources or payments, multiple nCDNs could form coalitions and modify their content allocation, if cooperation is beneficial for the members of the coalitions. In such a scenario, network operators would need distributed algorithms to compute and distribute the gains from cooperation among different nCDNs, so that the global content allocation reached through cooperation is stable. Therefore, two interesting questions are whether a cooperative solution exists in this scenario, and whether it is feasible for the network operators to implement it.

Bibliography

[1] *Netflix*. URL: `http://www.netflix.com`.

[2] Sandvine. *Global Internet Phenomena Report: 2H 2014*. Tech. rep. 2014.

[3] Cisco. *Visual Networking Index: Forecast and Methodology*. Tech. rep. 2015.

[4] MarketsandMarkets. *Content Delivery Network (CDN) Market by Solutions - Global Forecast to 2020*. Tech. rep. TC 2346. 2015, p. 157.

[5] *AT&T CDN Service*. URL: `http://www.business.att.com/enterprise/Service/hosting-services/content-delivery/distribution/`.

[6] *HP SpeedVideo CDN*. URL: `http://www.hp.com/go/cdn`.

[7] *Level3 Content Delivery Network*. URL: `http://www.level3.com/en/products-and-services/data-and-internet/cdn-content-delivery-network/`.

[8] *Alcatel-Lucent Velocix CDN*. URL: `http://resources.alcatel-lucent.com/?cid=165199`.

[9] AT&T Press Room. *Akamai and AT&T Forge Global Strategic Alliance to Provide Content Delivery Network Solutions*. 2012. URL: `http://www.att.com/gen/press-room?pid=23598%5C&cdvn=news%5C&newsarticleid=35788%5C&mapcode=`.

[10] Akamai. *Aura Managed CDN*. Aug. 2014. URL: `https://www.akamai.com/us/en/solutions/products/network-operator/managed-cdn-solutions.jsp`.

[11] EdgeCast. *Licensed CDN*. URL: `http://www.edgecast.com/solutions/licensed-cdn/`.

[12] Akamai. *Aura Licensed CDN*. 2015. URL: `https://www.akamai.com/us/en/solutions/products/network-operator/licensed-cdn-solutions.jsp`.

[13] AT&T. *AT&T Inc. Annual Report*. Tech. rep. 2014, pp. 1–81.

[14] *Hulu.* URL: http://www.hulu.com/.

[15] *Amazon.* URL: http://www.amazon.com/.

[16] Eric Schurman and Jake Brutlag. *The User and Business Impact of Server Delays, Additional Bytes, and HTTP Chunking in Web Search.* Tech. rep. Amazon and Google, 2009.

[17] Ron Kohavi and Roger Longbotham. "Online Experiments: Lessons Learned". In: *IEEE Computer* 40.9 (2007), pp. 85–87.

[18] Gonca Gursun, Mark Crovella, and Ibrahim Matta. "Describing and forecasting video access patterns". In: *Proc. of IEEE INFOCOM Mini Conference.* 2011, pp. 16–20.

[19] Gabor Szabo and Bernardo A. Huberman. "Predicting the popularity of online content". In: *Communications of the ACM* 53.8 (2010), p. 80.

[20] Huawei. *SingleEPC.* URL: http://www.huawei.com/.

[21] *Ipoque.* URL: http://www.ipoque.com/.

[22] Radware. *AppDirector.* URL: http://www.radware.com/.

[23] Frank Lehrieder et al. "The Impact of Caching on BitTorrent-Like Peer-to-Peer Systems". In: *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P).* 2010, pp. 1–10.

[24] Frank Lehrieder et al. "Caching for BitTorrent-Like P2P Systems: A Simple Fluid Model and Its Implications". In: *IEEE/ACM Transactions on Networking* 20.4 (2012), pp. 1176–1189.

[25] Tim Wu. "Network Neutrality, Broadband Discrimination". In: *Journal of Telecommunications and High Technology Law* 2 (2003), p. 141.

[26] Federal Communications Commission. *FCC Adopts Strong, Sustainable Rules To Protect The Open Internet.* Press Release. Feb. 2015. URL: https://www.fcc.gov/document/fcc-adopts-strong-sustainable-rules-protect-open-internet.

[27] European Commission. *Roaming charges and open Internet: questions and answers.* Press Release. June 2015. URL: http://europa.eu/rapid/press-release%5C_MEMO-15-5275%5C_en.htm.

[28] Terence Kelly and Dan Reeves. "Optimal Web cache sizing: Scalable methods for exact solutions". In: *Computer Communications* 24 (2001), pp. 163–173.

[29] Byung-Gon Chun et al. "Selfish caching in distributed systems: a game-theoretic analysis". In: *Proc. of ACM symposium on Principles of Distributed Computing (PODC).* 2004, pp. 21–30.

[30] Pablo Rodriguez, Christian Spanner, and Ernst W. Biersack. "Analysis of Web caching architectures: hierarchical and distributed caching". In: *IEEE/ACM Transactions on Networking* 9.4 (2001), pp. 404–418.

[31] Syam Gadde, Jeff Chase, and Michael Rabinovich. "Web caching and content distribution: a view from the interior". In: *Computer Communications* 24.2 (2001), pp. 222–231.

[32] *CDNetworks*. URL: `http://www.cdnetworks.com/`.

[33] *Akamai*. URL: `http://www.akamai.com`.

[34] *CloudFlare*. URL: `http://www.cloudflare.com/`.

[35] *Amazon CloudFront*. URL: `http://aws.amazon.com/cloudfront/`.

[36] *OverCache P2P*. URL: `http://www.oversi.com`.

[37] *PeerApp UltraBand*. URL: `http://www.peerapp.com`.

[38] S.-H. Gary Chan and Fouad A. Tobagi. "Modeling and dimensioning hierarchical storage systems for low-delay video services". In: *IEEE Transactions on Computers* 52.7 (2003), pp. 907–919.

[39] Arif Merchant, Qiang Ren, and Bhaskar Sengupta. "Hierarchical storage servers for video on demand: feasibility, design and sizing". In: *Proc. of IEEE GLOBECOM*. Vol. 1. 1996, pp. 272–278.

[40] Nikolaos Laoutaris, Vassilios Zissimopoulos, and Ioannis Stavrakakis. "On the Optimization of Storage Capacity Allocation for Content Distribution". In: *Computer Networks* 47.3 (2005), pp. 409–428.

[41] Alfred V. Aho, Peter J. Denning, and Jeffrey D. Ullman. "Principles of Optimal Page Replacement". In: *Journal of the ACM* 18.1 (1971), pp. 80–93.

[42] Oleg I. Aven, Edward G. Coffman Jr., and Yakov A. Kogan. *Stochastic Analysis of Computer Storage*. Kluwer Academic Publishers, 1987.

[43] Pei Cao and Sandy Irani. "Cost-aware WWW proxy caching algorithms". In: *Proc. of USENIX on Internet Technologies and Systems*. December. 1997, pp. 193–206.

[44] Omri Bahat and Armand M. Makowski. "Optimal replacement policies for non-uniform cache objects with optional eviction". In: *Proc. of IEEE INFOCOM*. 2003, pp. 427–437.

[45] Shudong Jin and Azer Bestavros. *Temporal Locality in Web Request Streams: Sources*. Tech. rep. 1999.

[46] *Squid Internet Object Cache*. URL: `http://www.squid-cache.org/`.

[47] Hao Che, Ye Tung, and Zhijun Wang. "Hierarchical Web caching systems: modeling, design and experimental results". In: *IEEE Journal on Selected Areas in Communications (JSAC)* 20.7 (2002), pp. 1305–1314.

[48] Emilio Leonardi and Giovanni L. Torrisi. "Least Recently Used caches under the Shot Noise Model". In: *Proc. of IEEE INFOCOM*. 2015, pp. 1–11.

[49] Donghee Lee et al. "LRFU: a spectrum of policies that subsumes the least recently used and least frequently used policies". In: *IEEE Transactions on Computers* 50.12 (2001), pp. 1352–1361.

[50] Martin Arlitt et al. "Evaluating content management techniques for Web proxy caches". In: *Proc. of Internet Server Performance*. 1999, pp. 1–9.

[51] Nimrod Megiddo and D. S. Modha. "Arc: A self-tuning, low overhead replacement cache". In: *Proc. of USENIX FAST*. 2003, pp. 115–130.

[52] Moinuddin K. Qureshi et al. "Adaptive insertion policies for high performance caching". In: *Proc. of International Symposium on Computer Architecture (ISCA)*. 2007, pp. 381–391.

[53] Andrea Araldo et al. "Cost-aware caching: optimizing cache provisioning and object placement in ICN". In: *Proc. of IEEE GLOBECOM*. 2014, pp. 1108–1113.

[54] György Dan and Niklas Carlsson. "Dynamic Content Allocation for Cloud-assisted Service of Periodic Workloads". In: *Proc. of IEEE INFOCOM*. 2014, pp. 1–9.

[55] Shudong Jin and Azer Bestavros. "Popularity-Aware GreedyDual-Size Web Proxy Caching Algorithms". In: *Proc. of Distributed Computing Systems (ICDCS)*. 2000, pp. 245–261.

[56] Di Niu et al. "Demand forecast and performance prediction in peer-assisted on-demand streaming systems". In: *Proc. of IEEE INFOCOM*. 2011, pp. 421–425.

[57] Di Niu, Chen Feng, and Baochun Li. "Pricing cloud bandwidth reservations under demand uncertainty". In: *Proc. of ACM SIGMETRICS Performance Evaluation Review*. 2012, pp. 151–162.

[58] Dilip K. Krishnappa et al. "On the feasibility of prefetching and caching for online TV services: a measurement study on hulu". In: *Proc. of International Conference on Passive and Active Measurement (PAM)*. 2011, pp. 72–80.

[59] Vijay V. Vazirani. *Approximation algorithms*. Springer-Verlag Berlin Heidelberg, 2003.

[60] Eric Cronin et al. "Constrained mirror placement on the Internet". In: *IEEE Journal on Selected Areas in Communications* 20.7 (2002), pp. 1369–1382.

[61] Lili Qiu, Venkata Padmanabhan, and Geoffrey Voelker. "On the placement of web server replicas". In: *Proc. of IEEE INFOCOM*. 2001, pp. 1587–1596.

[62] Elisha J. Rosensweig, Jim Kurose, and Don Towsley. "Approximate Models for General Cache Networks". In: *Proc. of IEEE INFOCOM*. 2010, pp. 1–9.

[63] Giovanna Carofiglio et al. "Modeling data transfer in content-centric networking". In: *Proc. of International Teletraffic Congress (ITC)*. 2011, pp. 111–118.

[64] Elisha J. Rosensweig, Daniel S. Menasche, and Jim Kurose. “On the Steady-State of Cache Networks”. In: *Proc. of IEEE INFOCOM*. 2013, pp. 887–895.

[65] Valentina Martina, Michele Garetto, and Emilio Leonardi. “A unified approach to the performance analysis of caching systems”. In: *Proc. of IEEE INFOCOM*. 2014, pp. 2040–2048.

[66] Michele Garetto, Emilio Leonardi, and Stefano Traverso. “Efficient analysis of caching strategies under dynamic content popularity”. In: *Proc. of IEEE INFOCOM*. 2015, pp. 2263–2271.

[67] Nikolaos Laoutaris, Hao Che, and Ioannis Stavrakakis. “The LCD Interconnection of LRU Caches and its Analysis”. In: *Performance Evaluation* 63.7 (2006), pp. 609–634.

[68] David Applegate et al. “Optimal content placement for a large-scale VoD system”. In: *Proc. of Co-NEXT*. 2010, pp. 1–12.

[69] Ivan Baev, Rajmohan Rajaraman, and Chaitanya Swamy. “Approximation Algorithms for Data Placement Problems”. In: *SIAM Journal on Computing* 38.4 (2008), pp. 1411–1429.

[70] Baruch Awerbuch, Yair Bartal, and Amos Fiat. “Distributed Paging for General Networks”. In: *Journal of Algorithms* 28.1 (1998), pp. 67–104.

[71] Negin Golrezaei et al. “FemtoCaching : Wireless Video Content Delivery through Distributed Caching Helpers”. In: *Proc. of IEEE INFOCOM*. 2012, pp. 1107–1115.

[72] Konstantinos Poularakis, George Iosifidis, and Leandros Tassiulas. “Approximation Algorithms for Mobile Data Caching in Small Cell Networks”. In: *IEEE Transactions on Communications* 62.10 (2014), pp. 3665–3677.

[73] Ulrich Berger. “Brown’s original fictitious play”. In: *Journal of Economic Theory* 135.1 (2007), pp. 572–578.

[74] Bary S.R. Pradelski and H. Peyton Young. “Learning efficient Nash equilibria in distributed systems”. In: *Games and Economic Behavior* 75.2 (2012), pp. 882–897.

[75] H. Peyton Young. “Learning by trial and error”. In: *Games and Economic Behavior* 65.2 (2009), pp. 626–643.

[76] Leonardo Boncinelli and Paolo Pin. “Stochastic Stability in Best Shot Network Games”. In: *Games and Economic Behavior* 75.2 (2012), pp. 538–554.

[77] Gerasimos Pollatos, Orestis Telelis, and Vassilis Zissimopoulos. “On the social cost of distributed selfish content replication”. In: *Proc. of IFIP Networking*. 2008, pp. 195–206.

[78] Eva Jaho, Merkourios Karaliopoulos, and Ioannis Stavrakakis. “Social similarity favors cooperation: the distributed content replication case”. In: *IEEE Transactions on Parallel and Distributed Systems* 24.3 (2013), pp. 601–613.

[79] Sem Borst, Varun Gupta, and Anwar Walid. “Distributed Caching Algorithms for Content Distribution Networks”. In: *Proc. of IEEE INFOCOM.* 2010, pp. 1478–1486.

[80] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. “Placement Algorithms for Hierarchical Cooperative Caching”. In: *Journal of Algorithms* 38.1 (2001), pp. 260–302.

[81] Madhukar R. Korupolu and Michael Dahlin. “Coordinated placement and replacement for large-scale distributed caches”. In: *IEEE Transactions on Knowledge and Data Engineering* 14.6 (2002), pp. 1317–1329.

[82] Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. “An Analysis of Approximations for Maximizing Submodular Set Functions - II”. In: *Mathematical Programming Study* 8 (1978), pp. 73–78.

[83] David Karger et al. “Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web”. In: *Proc. of ACM Symposium on Theory of Computing (STOC)* (1997), pp. 654–663.

[84] Utpal Paul et al. “Understanding traffic dynamics in cellular data networks”. In: *Proc. of IEEE INFOCOM.* 2011, pp. 882–890.

[85] Balaji Srivastava, Shekhar Krithikaivasan et al. “Benefits of Traffic Engineering using QoS Routing Schemes and Network Controls”. In: *Proc. of Computer Communications.* 2002, pp. 271–275.

[86] Peter Eckersley, Fred von Lohmann, and Seth Schoen. *Packet Forgery By ISPs: A Report on the Comcast Affair.* White paper. Nov. 2007.

[87] BEREC. *A view of traffic management and other practices resulting in restrictions to the open Internet in Europe.* Tech. rep. 2012, pp. 1–39.

[88] Daniel O. Awduche et al. *Overview and Principles of Internet Traffic Engineering.* RFC. Internet Engineering Task Force (IETF), 2002. URL: `http://www.ietf.org/rfc/rfc3272.txt`.

[89] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. “Traffic engineering with traditional IP routing protocols”. In: *IEEE Communications Magazine* 40.10 (2002), pp. 118–124.

[90] Ryan S. Peterson and Emin Gün Sirer. “Antfarm : Efficient Content Distribution with Managed Swarms”. In: *Proc. of NSDI.* 2009, pp. 107–122.

[91] Ryan S. Peterson, Bernard Wong, and Emin Gün Sirer. “A Content Propagation Metric for Efficient Content Distribution”. In: *Proc. of ACM SIGCOMM.* Vol. 41. 4. New York, New York, USA, 2011, pp. 326–337.

[92] Abhigyan Sharma, Arun Venkataramani, and Antonio A. Rocha. “Pros & cons of model-based bandwidth control for client-assisted content delivery”. In: *Proc. of Communication Systems and Networks (COMSNETS).* 2014, pp. 1–8.

[93] *Content Delivery Networks Interconnection (CDNI).* Working Group. Internet Engineering Task Force (IETF). URL: `http://datatracker.ietf.org/wg/cdni/`.

[94] Scott Puopolo et al. *Content Delivery Network (CDN) Federations - How SPs Can Win the Battle for Content-Hungry Consumers.* White Paper. Cisco, 2011, pp. 1–9.

[95] Vytautas Valancius et al. "Quantifying the benefits of joint content and network routing". In: *Proc. of ACM SIGMETRICS Performance Evaluation Review.* 2013, pp. 243–254.

[96] Al-Mukaddim K. Pathan and Rajkumar Buyya. *A Taxonomy and Survey of Content Delivery Networks.* Tech. rep. The University of Melbourne, 2007, pp. 1–44.

[97] Gilles Bertrand et al. *Use Cases for Content Delivery Network Interconnection.* RFC. Internet Engineering Task Force (IETF), 2012. URL: `http://tools.ietf.org/html/rfc6770`.

[98] Yannick Le Louedec et al. *Final requirements for Open Content Aware Networks.* Deliverable D2.2. FP7 - Open ContEnt Aware Networks (OCEAN), 2013.

[99] Marc Latouche et al. *The CDN Federation - Solutions for SPs and Content Providers To Scale a Great Customer Experience.* Tech. rep. October. Cisco, 2012, pp. 1–11.

[100] Larry Peterson, Bruce Davie, and Ray van Brandenburg. *Framework for Content Distribution Network Interconnection (CDNI).* RFC. Internet Engineering Task Force (IETF), 2014.

[101] Jan Seedorf et al. *CDNI Request Routing: Footprint and Capabilities Semantics.* Draft. Internet Engineering Task Force (IETF), 2015.

[102] Ben Niven-Jenkins and Ray van Brandenburg. *Request Routing Redirection Interface for CDN Interconnection.* Draft. Internet Engineering Task Force (IETF), 2015.

[103] Kent Leung et al. *URI Signing for CDN Interconnection (CDNI).* Draft. Internet Engineering Task Force (IETF), 2015.

[104] Ben Niven-Jenkins et al. *CDN Interconnection Metadata.* Draft. Internet Engineering Task Force (IETF), 2015.

[105] John F. Nash. "Equilibrium points in n-person games". In: *Proc. of the National Academy of Sciences of the United States of America* 36.1 (1950), pp. 48–49.

[106] Ragavendran Gopalakrishnan et al. "Cache me if you can: capacitated selfish replication games". In: *Proc. of LATIN.* Ed. by David Fernández-Baca. Vol. 7256. Lecture Notes in Computer Science. Berlin, Heidelberg, 2012, pp. 420–432.

[107] Igal Milchtaich. “Congestion games with player-specific payoff functions”. In: *Games and Economic Behavior* 13.1 (1996), pp. 111–124.

[108] Nikolaos Laoutaris et al. “Distributed selfish replication”. In: *IEEE Transactions on Parallel and Distributed Systems* 17.12 (2006), pp. 1401–1413.

[109] György Dán. “Cache-to-cache: Could ISPs cooperate to decrease peer-to-peer content distribution costs?” In: *IEEE Transactions on Parallel and Distributed Systems* 22.9 (2011), pp. 1469–1482.

[110] Vittorio Bilò et al. “Graphical Congestion Games”. In: *Algorithmica* 61.2 (2011), pp. 274–297.

[111] Dimitris Fotakis et al. “The Impact of Social Ignorance on Weighted Congestion Games”. In: *Proc. of WINE*. 2009, pp. 316–327.

[112] Van Jacobson et al. “Networking named content”. In: *Proc. of ACM CoNEXT*. 2009.

[113] Teemu Koponen et al. “A data-oriented (and beyond) network architecture”. In: *Proc. of ACM SIGCOMM*. Vol. 37. 4. 2007, pp. 181–192.

[114] Christian Dannewitz. “NetInf: An Information-Centric Design for the Future Internet”. In: *Proc. of GI/TG KuVS Work. on The Future Internet*. 2009.

[115] Antonio Carzaniga, Matthew J Rutherford, and Alexander L Wolf. “A Routing Scheme for Content-Based Networking”. In: *Proc. of IEEE INFOCOM*. 2004, pp. 1–11.

[116] Elisha J. Rosensweig and Jim Kurose. “Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks”. In: *Proc. of IEEE INFOCOM*. 2009, pp. 2631–2635.

[117] Vasilis Sourlas, Paris Flegkas, and Leandros Tassiulas. “A novel cache aware routing scheme for Information-Centric Networks”. In: *Computer Networks* 59 (2014), pp. 44–61.

[118] Ioannis Psaras et al. “Modelling and evaluation of CCN-caching trees”. In: *Proc. of IFIP Networking*. 2011, pp. 78–91.

[119] Dario Rossi and Giuseppe Rossini. “On sizing CCN content stores by exploiting topological information”. In: *Proc. of IEEE INFOCOM, NOMEN Workshop*. 2012, pp. 280–285.

[120] Vasilis Sourlas et al. “Distributed Cache Management in Information-Centric Networks”. In: *IEEE Transactions on Network and Service Management* 10.3 (2013), pp. 286–299.

[121] Ioannis Psaras, Wei Koong Chai, and George Pavlou. “In-Network Cache Management and Resource Allocation for Information-Centric Networks”. In: *IEEE Transactions on Parallel and Distributed Systems* 25.11 (2014), pp. 2920–2931.

[122] Duane Wessels and K. Claffy. *Internet Cache Protocol (ICP), version 2.* RFC. Internet Engineering Task Force (IETF), 1997, pp. 1–8. URL: `https://www.ietf.org/rfc/rfc2186.txt`.

[123] Li Fan et al. "Summary cache: a scalable wide-area Web cache sharing protocol". In: *IEEE/ACM Transactions on Networking* 8.3 (2000), pp. 281–293.

[124] Alex Rousskov and Duane Wessels. "Cache Digests". In: *Proc. of Computer Networks and ISDN Systems.* 1998, pp. 22–41.

[125] Sandra G. Dykes and Kay A. Robbins. "Limitations and Benefits of Cooperative Proxy Caching". In: *IEEE Journal on Selected Areas in Communications* 20.7 (2002), pp. 1290–1304.

[126] Alec Wolman et al. "On the scale and performance of cooperative Web proxy caching". In: *Proc. of ACM Symposium on Operating Systems Principles (SOSP).* Vol. 33. 5. 1999, pp. 16–31.

[127] György Dan. "Cooperative caching and relaying strategies for peer-to-peer content delivery". In: *Proc. of the International Workshop on Peer-to-Peer Systems (IPTPS).* 2008, pp. 1–16.

[128] Mohamed Hefeeda and Behrooz Noorizadeh. "On the Benefits of Cooperative Proxy Caching for Peer-to-Peer Traffic". In: *IEEE Transactions on Parallel and Distributed Systems* 21.7 (2010), pp. 998–1010.

[129] Zhe Li and Gwendal Simon. "Cooperative Caching in a Content Centric Network for Video Stream Delivery". In: *Journal of Network and Systems Management* 23.3 (2015), pp. 445–473.

[130] Emil Eriksson, Valentino Pacifici, and György Dán. "Efficient Distribution of Visual Processing Tasks in Multi-camera Visual Sensor Networks". In: *Workshop on Distributed and Cooperative Visual Representation and Analysis (DCVRA).* 2015, pp. 1–6.

[131] Valentino Pacifici and György Dan. "Stable Content-peering of Autonomous Systems in a Content-centric Network". In: *Proc. of Swedish National Computer Networking Workshop (SNCNW).* 2013, pp. 1–4.

[132] Valentino Pacifici and György Dán. "Content-peering Dynamics of Autonomous Caches in a Content-centric Network". In: *Proc. of IEEE INFOCOM.* 2013, pp. 1079–1087.

[133] Valentino Pacifici, Frank Lehrieder, and György Dán. "Cache Capacity Allocation for BitTorrent-Like Systems to Minimize Inter-ISP Traffic". In: *Proc. of IEEE INFOCOM.* 2012, pp. 1512–1520.

[134] Valentino Pacifici and György Dan. "A Game Theoretic Analysis of Selfish Content Replication on Graphs". In: *Poster presented at IEEE INFOCOM* (2012).

[135] Valentino Pacifici and György Dán. "Selfish Content Replication on Graphs". In: *Proc. of the 23rd International Teletraffic Congress.* 2011, pp. 119–126.

[136] Valentino Pacifici, Frank Lehrieder, and György Dan. “On the Benefits of P2P Cache Capacity Allocation”. In: *Poster presented at the International Teletraffic Congress (ITC)* (2011), pp. 312–313.

[137] Valentino Pacifici and György Dan. “A Game Theoretic Analysis of Selfish Content Replication on Graphs”. In: *Proc. of Swedish National Computer Networking Workshop (SNCNW)*. 2011, pp. 1–4.

Paper **A**

Distributed Algorithms for Content Caching in Mobile Backhaul Networks

Valentino Pacifici and Slađana Jošilo and György Dán

submitted to IFIP Networking 2016 (NETWORKING), 2016.

Distributed Algorithms for Content Caching in Mobile Backhaul Networks

Valentino Pacifici and Slađana Jošilo and György Dán
ACCESS Linnaeus Center, School of Electrical Engineering
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: {pacifici, josilo, gyuri}@kth.se

Abstract

The growing popularity of mobile multimedia content and the increase of wireless access bitrates are straining backhaul capacity in mobile networks. A cost-effective solution to reduce the strain, enabled by emerging all-IP 4G and 5G mobile backhaul architectures, could be in-network caching of popular content during times of peak demand. In this paper we formulate the problem of content caching in a mobile backhaul as a binary integer programming problem, and we propose a 2-approximation algorithm for the problem. The 2-approximation requires full information about the network topology and the link costs, as well as about the content demands at the different caches, we thus propose two distributed algorithms that are based on local information on the content demands. We show that the distributed algorithms terminate in a finite number of steps, and we provide analytical results on their approximation ratios. We use simulations to evaluate the proposed algorithms in terms of the achieved approximation ratio and computational complexity on realistic mobile backhaul topologies.

1 Introduction

The penetration of high speed mobile access technologies, such as HSDPA and LTE, together with the proliferation of powerful handheld devices has stimulated a rapid increase of user demand for mobile multimedia content in recent years. The traffic growth is predicted to continue in coming years, with an estimated 10-fold increase in mobile data traffic in 5 years and an increasing peak-to-average traffic ratio, and puts significant strain on mobile backhaul capacity.

Recent measurement studies of mobile data traffic indicate that caching could be an effective means of decreasing the mobile backhaul bandwidth requirements: caching could reduce the bandwidth demand by up to 95% during peak hours and could at the same time reduce content delivery time by a factor of three [1]. At the same time, mobile traffic is dominated by downloads; up to 75% of daily traffic load comes from download traffic, and the demand shows significant diurnal fluctuations with low loads during early morning hours [2].

While tunelling imposed by previous 3GPP standards made backhaul in-network caching technically challenging, allowing only caches at the network edge, in emerging all-IP mobile backhaul architectures the caches could be co-located with every switch and could implement cooperative caching policies throughout the backhaul. Since fairly accurate content popularity predictions can be obtained for Web and video content [3, 4], the most popular contents could be downloaded into the caches of the mobile backhaul in the early morning hours when the load is relatively low, thereby alleviating the traffic demand during peak hours.

Given predicted content popularities, a fundamental problem of in-network caching in a mobile backhaul is to find efficient content placement algorithms that take into consideration the characteristics of mobile backhaul topologies and of mobile data traffic. The algorithms should achieve close to optimal bandwidth cost savings and should have low computational complexity. Furthermore, they should require as little information as possible, e.g., about content popularities and network topology, in order to allow fully distributed operation and scaling to large topologies with small communication overhead. While previous works proposed centralized and distributed content placement algorithms for two-level hierarchical topologies [5], general topologies with an ultrametric [6], and topologies in a metric space [7], efficient distributed algorithms based on limited topological information have received little attention.

In this paper we formulate the problem of content placement in a mobile backhaul based on predicted demands as a 0-1 integer programming problem. We show that a 2-approximation to the problem can be obtained using a distributed greedy algorithm when global information is available, and propose two computationally simple distributed algorithms that do not require global information. We evaluate the algorithms through extensive simulations on various network topologies. Our results show that local information about object demands is not sufficient for achieving good performance, but the proposed h-Push Down algorithm achieves consistently good performance based on a limited amount of information about object placements.

The rest of the paper is organized as follows. Section 2 describes the system model and provides the problem formulation. Section 3 describes the 2-approximation algorithm based on global information, and Section 4 describes the distributed algorithms based on local information. Section 5 shows performance results based on simulations. Section 6 discusses related work and Section 7 concludes the paper.

2 System Model and Problem Formulation

We consider a typical mobile backhaul, and model its active topology by a symmetric acyclic directed graph $\mathcal{G}(\mathcal{N}, E)$, where the vertices $\mathcal{N}$ are routers that connect cell sites and may aggregate traffic from other routers (and thus cell cites), and for every connected pair of nodes $i, j \in \mathcal{N}$ there exist edges $(i,j) \in E$ and $(j,i) \in E$. Observe that since $\mathcal{G}$ is connected and acyclic, $\mathcal{G}$ is a tree. We denote by $\mathcal{L}$ the set of leaf nodes in $\mathcal{G}$, by $\mathcal{I}$ the set of internal nodes and by n_0 the root node, i.e., $\mathcal{N} = \mathcal{L} \cup \mathcal{I} \cup n_0$. We denote the unique simple path from node i to node j by

$$P_{i,j} = \big((i, v_1), (v_1, v_2), ..., (v_{|P_{i,j}|-1}, j)\big)\,, \tag{1}$$

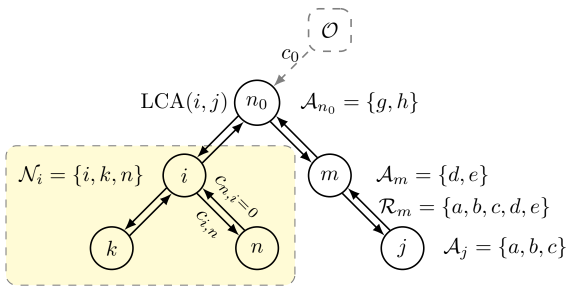
and we denote by $|P_{i,j}|$ the number of edges in path $P_{i,j}$. Observe that $|P_{i,j}| = |P_{j,i}|$. We define the level $l(i)$ of node $i \in \mathcal{N}$ in the tree $\mathcal{G}$ as the number of edges from node i to the tree's root node n_0 in the unique simple path from i to n_0, i.e., $l(i) = |P_{i,n_0}|$. We denote the children of node $i \in \mathcal{N}$ by $\mathcal{C}(i) \triangleq \{j|(i,j) \in E \wedge l(j) > l(i)\}$ and the parent of node i by $\mathcal{P}(i)$, where $\mathcal{P}(i) \in \mathcal{N}$ such that $i \in \mathcal{C}(\mathcal{P}(i))$. We denote by $\mathcal{P}^l(i)$ the l^{th}-ancestor of node i, e.g., $\mathcal{P}^2(i) = \mathcal{P}(\mathcal{P}(i))$. By definition $\mathcal{P}^0(i) = i$. We refer to an edge (i,j) as the downlink direction if $j \in \mathcal{C}(i)$ and as the uplink direction if $i \in \mathcal{C}(j)$.

We say that two nodes are siblings if they have the same parent, and define the sibling set $\mathcal{S}(i) \triangleq \{j|\mathcal{P}(j) = \mathcal{P}(i) \wedge i \neq j\}$. We denote the descendants of node i by $\mathcal{D}(i) \triangleq \{j|l(j) > l(i) \wedge \text{LCA}(i,j) = i\}$, where $\text{LCA}(i,j)$ denotes the lowest common ancestor of nodes i and j, furthermore we use the notation $\mathcal{G}_i(\mathcal{N}_i, E_i)$ for the subgraph induced by $\mathcal{N}_i = \{i\} \cup \mathcal{D}(i)$ rooted in i.

2.1 Objects, Demand and Storage

We denote the set of objects requested by mobile nodes by $\mathcal{O}$. We follow common practice and consider that every object has unit size [8, 9], which is a reasonable simplification if content is divisible into unit-sized chunks. We denote the average request rate (demand) predicted for the peak hours for object $o \in \mathcal{O}$ at the cell site connected to node i by w_i^o.

Every node $i \in \mathcal{N}$ is equipped with a cache, and we denote the size of the cache at node i by K_i. We denote the set of objects stored in the cache at node i by $\mathcal{A}_i \subset \mathcal{O}$, $|\mathcal{A}_i| \leq K_i$. We use the shorthand notation $\mathcal{A}_V \triangleq (\mathcal{A}_j)_{j \in V}$, where $V \subseteq \mathcal{N}$, and $\mathcal{A}_{-i} \triangleq (\mathcal{A}_j)_{j \in \mathcal{N}\setminus\{i\}}$. We denote by $\mathfrak{A}_i$ the set of object placements that satisfy the storage capacity constraint at node i, i.e. $\mathfrak{A}_i = \{\mathcal{A} \in 2^{\mathcal{O}} : |\mathcal{A}| \leq K_i\}$, where $2^{\mathcal{O}}$ is the powerset of $\mathcal{O}$. Finally, we denote the set of objects stored at node i and at its descendants by $\mathcal{R}_i(\mathcal{A}) = \mathcal{A}_i \bigcup_{j \in \mathcal{D}(i)} \mathcal{R}_j(\mathcal{A})$. Figure 1 shows an example topology with a maximum level of 2, illustrating some of the commonly used notation.



$\mathcal{O}$
c_0
LCA(i,j)
n_0
$\mathcal{A}_{n_0} = \{g,h\}$
$\mathcal{N}_i = \{i,k,n\}$
i
$c_{n,i}=0$
$c_{i,n}$
m
$\mathcal{A}_m = \{d,e\}$
$\mathcal{R}_m = \{a,b,c,d,e\}$
k
n
j
$\mathcal{A}_j = \{a,b,c\}$

Figure 1: Example backhaul topology with nodes in three levels, showing commonly used notation.

2.2 Cost model

We denote the unit cost of using edge (i,j) by $c_{i,j}$. Since during peak hours most of the traffic in a mobile backhaul is flowing downlink (serving users' requests for content) [1, 2], we consider that uplink edges have zero unit cost, i.e., $c_{i,\mathcal{P}(i)} = 0$. Without loss of generality, the cost of downlink edges is $c_{\mathcal{P}(i),i} > 0$. We consider that edge costs are additive, i.e., if a request for object o arrives at node i and is served from node j then the unit cost is

$$d_{i,j} = \sum_{(v,w)\in P_{j,i}} c_{v,w}. \tag{2}$$

We call $d_{i,j}$ the *distance* from node j to node i. Note that the terms $c_{v,w}$ in (2) are zero if they correspond to an uplink, i.e., if $w = \mathcal{P}(v)$. Furthermore, observe that in general $d_{j,i} \neq d_{i,j}$, thus distance is not symmetric (hence it is a hemimetric).

A request for object o generated by a mobile user connected to the cell site at node $i \in \mathcal{N}$ is served locally if $o \in \mathcal{A}_i$. Otherwise, if node i has a descendant $j \in \mathcal{D}(i)$ for which $o \in \mathcal{A}_j$, the node forwards the request to the closest such descendant. Otherwise, node i forwards the request to its parent $\mathcal{P}(i)$, which follows the same algorithm for serving the request. If an object o is not stored in any node (i.e., $o \notin \mathcal{R}_{n_0}$) then it needs to be retrieved through the Backbone via the root node n_0 at a unit cost of c_0.

Given a placement $\mathcal{A} = (\mathcal{A}_j)_{j\in\mathcal{N}}$ we can define the unit cost to serve a request for object o at node i as

$$d_i(o,\mathcal{A}) = \begin{cases} \min\limits_{\{j\in\mathcal{N}|o\in\mathcal{A}_j\}} d_{i,j} & \text{if } o \in \mathcal{R}_{n_0} \\ d_{i,n_0} + c_0 & \text{if } o \notin \mathcal{R}_{n_0}, \end{cases} \tag{3}$$

which together with the demand w_i^o determines the cost incurred by node i as

$$C_i(\mathcal{A}) = \sum_{o\in\mathcal{O}} C_i^o(\mathcal{A}) = \sum_{o\in\mathcal{O}} w_i^o d_i(o,\mathcal{A}). \tag{4}$$

Finally, we define the total cost $C(\mathcal{A}) = \sum_{i\in\mathcal{N}} C_i(\mathcal{A})$.

2.3 Problem formulation

Motivated by minimizing the congestion in the mobile backhaul during peak hours, our objective is to find a placement that minimizes the total cost $C(\mathcal{A})$. We refer to this as the mobile backhaul content placement problem (MBCP), which can be formulated as finding $\bar{\mathcal{A}} = \arg\min_{\mathcal{A}\in\times_{i\in\mathcal{N}}\mathfrak{A}_i} C(\mathcal{A})$.

It is easy to see that the MBCP problem can be formulated as the following $0-1$ integer linear program

$$\begin{aligned}
\min \sum_{i\in\mathcal{N}}\sum_{o\in\mathcal{O}} w_i^o\Big(\sum_{j\in\mathcal{N},j\neq i} d_{i,j}x_{i,j,o} + (d_{i,n_0}+c_0)x_{i,-1,o}\Big) \quad \text{s.t.}&\\
\sum_{o\in\mathcal{O}} x_{i,o} \le K_i, &\quad \forall i\in\mathcal{N}\\
x_{i,j,o} \le x_{j,o}, &\quad \forall i,j\in\mathcal{N}, o\in\mathcal{O}\\
\sum_{j\in\mathcal{N}} x_{i,j,o} + x_{i,-1,o} \ge 1, &\quad \forall i\in\mathcal{N}, o\in\mathcal{O}\\
x_{i,o}, x_{i,j,o}, x_{i,-1,o} \in \{0,1\},&
\end{aligned}$$

where $x_{i,o}\in\{0,1\}$ indicates whether object o is in the storage of node i (i.e. $x_{i,o}=1 \Leftrightarrow o\in\mathcal{A}_i$), $x_{i,j,o}\in\{0,1\}$ indicates whether a request for object o at node i is served from node j, and $x_{i,-1,o}\in\{0,1\}$ indicates whether object o is retrieved from the Backbone, i.e., the *level* of the Backbone is indicated with -1.

Unfortunately, solving the MBCP problem is computationally infeasible for tens of thousands of objects and tens to hundreds of nodes. We are thus interested in finding computationally feasible, scalable distributed algorithms to approximate the solution.

3 Distributed 2-Approximation Algorithm Based On Global Information

In what follows we show that if global information is available about the object demands and placements at every node of the network, then it is possible to obtain a 2-approximation to the optimal solution using the *Depth First Greedy* (*DFG*) algorithm. The *DFG* algorithm is based on a depth-first traversal of the graph $\mathcal{G}$, i.e., an ordering $i_1,\dots,i_{|\mathcal{N}|}$ of the vertices in $\mathcal{N}$, and can be executed by the nodes in an iterative (distributed) manner. The algorithm starts with an empty allocation ($\mathcal{A}_i=\emptyset$); at iteration $1\le k\le|\mathcal{N}|$ node i_k populates its cache with K_{i_k} objects, one at a time, that provide the highest global cost saving. The *DFG* algorithm is shown in Figure 2.

DFG Algorithm

```
1: INPUT: DF Traversal $(i_1, \ldots, i_{|\mathcal{N}|})$
2: $k \leftarrow 1$
3: $\mathcal{A}_i \leftarrow \emptyset$, for all $i \in \mathcal{N}$
4: for $k = 1 \ldots |\mathcal{N}|$ do
5:     while $|\mathcal{A}_{i_k}| < K_{i_k}$ do
6:         $o^* \leftarrow \arg\max_{o \in \mathcal{O}}(C(\mathcal{A}_{-i_k}, \mathcal{A}_{i_k}) - C(\mathcal{A}_{-i_k}, \mathcal{A}_{i_k} \cup \{o\}))$
7:         $\mathcal{A}_{i_k} \leftarrow \mathcal{A}_{i_k} \cup \{o^*\}$
8:     end while
9: end for
```

Figure 2: Pseudo-code of the *DFG* algorithm

Theorem 1. *The* DFG *algorithm is a 2-approximation algorithm for the MBCP problem in terms of cost saving, i.e.,* $\frac{C(\bar{\mathcal{A}})-C(\emptyset)}{C(\mathcal{A}^{DFG})-C(\emptyset)} \leq 2.$.

Before we prove the theorem we introduce some definitions and previous results.

Definition 1. Let E be a finite set and let $\mathcal{F}$ be a collection of subsets of E. The pair $(E,\mathcal{F})$ is a *partition matroid* if $E = \bigcup_{i=1}^{k} E_i$ is the disjoint union of k sets, $l_1, ..., l_k$ are positive integers and $\mathcal{F} = \{F | F = \bigcup_{i=1}^{k} F_i, F_i \subseteq E_i, |F_i| \leq l_i, i = 1, ..., k\}$.

Definition 2. Let E be a finite set, and $f : 2^E \rightarrow \mathbb{R}$ a real valued function on subsets of E. Then f is submodular if for every $A, B \in E$ we have

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B).$$

Let us now recall a fundamental result about the maximization of submodular functions over partition matroids.

Lemma 1. *[10] Let $\mathcal{F}$ be a partition matroid over a set E, and $f : \mathcal{F} \rightarrow \mathbb{R}$ be a non-decreasing submodular function with $f(\emptyset) = 0$. Then the* DFG *algorithm achieves a 2-approximation of* $\max_{F \in \mathcal{F}} f(F)$.

In what follows we show that MBCP can be formulated as the maximization of a non-decreasing submodular function over a partition matroid. Let us define for every object $o \in \mathcal{O}$ one fictitious object (o, i) per node $i \in \mathcal{N}$, i.e., $(o, i) \in \mathcal{O} \times \mathcal{N}$. The set of fictitious objects that can be assigned to node i is then $\mathcal{E}_i = \{(o, i) | o \in \mathcal{O}\}$ and we define the set $\mathcal{E} = \bigcup_{i \in \mathcal{N}} \mathcal{E}_i$. We denote by $\mathfrak{A}$ the family of subsets of $\mathcal{E}$, defined as $\mathfrak{A} = \times_{i \in \mathcal{N}} \mathfrak{A}_i$, where $\mathfrak{A}_i \subseteq \mathcal{E}_i$, $|\mathfrak{A}_i| \leq K_i$ is the set of object placements that satisfy the storage capacity constraint at node i, as defined in Section 2.1.

Proposition 2. *The pair $(\mathcal{E}, \mathfrak{A})$ is a partition matroid.*

Proof. Consider an allocation $\mathcal{A} \in \mathfrak{A}$ and a fictitious object $(o, i) \in \mathcal{A}_i$. If we remove (o, i) from $\mathcal{A}_i$, i.e. $\mathcal{A}'_i = \mathcal{A}_i \setminus \{(o, i)\}$, then $\mathcal{A}'_i \subseteq \mathcal{E}_i$ will still hold as well as $\mathcal{A}_j \subseteq \mathcal{E}_j$, for $j \in \mathcal{N} \setminus \{i\}$, which implies that $(\mathcal{E}, \mathfrak{A})$ is an independence system.

Consider now two allocations $\mathcal{A}, \mathcal{A}' \in \mathfrak{A}$. If $|\mathcal{A}| < |\mathcal{A}'|$ then $\exists \mathcal{E}_i$ such that $|\mathcal{A}' \cap \mathcal{E}_i| > |\mathcal{A} \cap \mathcal{E}_i|$, which implies that there is a node $i \in \mathcal{N}$ with at least one free space in its cache, i.e. $|\mathcal{A}_i| < K_i$. Therefore, there is an $(o, i) \in (\mathcal{A}' \setminus \mathcal{A}) \cap \mathcal{E}_i$ such that $\mathcal{A} \cup \{(o, i)\} \in \mathfrak{A}$. □

Proof of Theorem 1. We prove the theorem by showing that the function $\bar{C}(\mathcal{A}) = -C(\mathcal{A})$ is a nondecreasing submodular function on $\mathcal{E}$. Let us define the change of the global cost after inserting an object o in the cache of node i as $\Delta C(\mathcal{A}) = \bar{C}(\mathcal{A} \cup \{(o, i)\}) - \bar{C}(\mathcal{A})$, where $\mathcal{A} \in \mathfrak{A}$ and $\exists i \in \mathcal{N}$ for which $|\mathcal{A}_i| < K_i$. We show that $\bar{C}(\mathcal{A} \cup (o, i)\}) - \bar{C}(\mathcal{A}) \geq \bar{C}(\mathcal{A}' \cup (o, i)\}) - \bar{C}(\mathcal{A}')$ for all $\mathcal{A} \subseteq \mathcal{A}' \in \mathfrak{A}$ and $(o, i) \in \mathcal{E}_i \setminus \mathcal{A}'_i$. We now distinguish between two cases. If $\exists j$ such that $(o, j) \in \mathcal{A}'_j \setminus \mathcal{A}_j$ then the difference $\Delta C(\mathcal{A})$ is

$$\Delta C(\mathcal{A}) = c_0 \sum_{k \in \{\mathcal{N} |\, \mathrm{LCA}(k,i) = n_0\}} w_k^o + (c_0 + d_{i,n_0}) \sum_{k \in \mathcal{N}_i} w_k^o + \sum_{t=1}^{l(i)-1} (c_0 + d_{\mathcal{P}^t(i), n_0}) \sum_{k \in \{\mathcal{N}_{\mathcal{P}^t(i)} \setminus \mathcal{N}_{\mathcal{P}^{t-1}(i)}\}} w_k^o,$$

and the difference $\Delta C(\mathcal{A}')$ is

$$\Delta C(\mathcal{A}') = (c_0 + d_{i,\mathrm{LCA}(j,i)}) \sum_{k \in \mathcal{N}_i} w_k^o + \sum_{t=1}^{l(i)-l(\mathrm{LCA}(j,i))-1} (c_0 + d_{\mathcal{P}^t(i), \mathrm{LCA}(j,i)}) \sum_{k \in \{\mathcal{N}_{\mathcal{P}^t(i)} \setminus \mathcal{N}_{\mathcal{P}^{t-1}(i)}\}} w_k^o.$$

Since $l(\mathrm{LCA}(j, i)) \geq 0$, it holds that $\Delta C(\mathcal{A}) > \Delta C(\mathcal{A}')$. Otherwise, if $\exists j$ such that $(o, j) \in \mathcal{A}_j$ or if $\nexists j$ such that $(o, j) \in \mathcal{A}'_j$ then $\Delta C(\mathcal{A}) = \Delta C(\mathcal{A}')$. The result then follows by applying Lemma 1 to $C(\emptyset) - C(\mathcal{A})$. □

Observe that the approximation ratio is bounded for arbitrary traversals of the graph. Nonetheless, a pre-order depth-first traversal allows for a distributed implementation of *DFG* with a communication overhead of $\sum_{k=1}^{|\mathcal{N}|} (|\mathcal{N}| - k) K_{i_k}$.

It is important to note that *DFG* differs from the distributed global greedy (*DGG*) algorithm used in [5, 11]. *DGG* chooses in every iteration the fictitious item (i, o) that maximizes the cost saving, and thus has computational complexity $O(|\mathcal{N}|^2 \max_i K_i O \log(|\mathcal{N}| O))$. In contrast, *DFG* populates the caches of the nodes one-by-one, and thus has computational complexity $O(|\mathcal{N}| \max_{i_k} K_{i_k} O \log(O))$.

4 Distributed Algorithms based on Local Information

In what follows we propose two distributed algorithms that do not need global information about the demands and the network topology.

4.1 Local Greedy Swapping (LGS) Algorithm

The first algorithm, called Local Greedy Swapping (LGS), allows nodes to swap objects with their parents based on the aggregate demands and the object placements in their *descendants only.* Denoting the placement at node i at iteration k by $\mathcal{A}_i(k)$, the LGS algorithm starts with an arbitrary initial object placement $(\mathcal{A}_i(0))_{i\in\mathcal{N}}$ in which each node $i \in \mathcal{N}$ stores K_i objects. At iteration k the algorithm computes the set of beneficial swaps $T(\mathcal{A}(k)) \subset \mathcal{N} \times \mathcal{O}^2$. A triplet $(i, o, p) \in T(\mathcal{A}(k))$ corresponds to that node i can swap object $p \in \mathcal{A}_i(k)$ with object $o \in \mathcal{A}_{\mathcal{P}(i)}(k)$ at its parent node $\mathcal{P}(i)$. For $i = n_0$, i.e., $(n_0, o, p) \in T(\mathcal{A}(k))$ the root node n_0 can evict object p and can fetch object o through the Backbone. The set of implemented swaps $S(\mathcal{A}(k)) \subseteq T(\mathcal{A}(k)$ is then chosen to increase the local cost saving greedily.

To define the set of beneficial swaps $T(\mathcal{A})$, let us introduce the function $I(i, o, p)$ to indicate whether the aggregate demand at node i and its descendants $\mathcal{D}(i)$ is higher for object o than for object p,

$$I(i,o,p)=\begin{cases}1, & \text{if} \sum_{j\in\mathcal{N}_i}(w_j^o - w_j^p) > 0\\ 0, & \text{otherwise.}\end{cases} \tag{5}$$

Given a placement $\mathcal{A}$, node i might be interested in swapping object $p \in \mathcal{A}_i$ with object $o \in \mathcal{A}_{\mathcal{P}(i)}$ at its parent if $I(i, o, p) = 1$ or if p is available in the cache of node i's descendants $\mathcal{D}(i)$, i.e., $p \in \mathcal{R}_i \setminus \mathcal{A}_i$, as in this case node i can retrieve object p at no cost even if $p \notin \mathcal{A}_i$. We use this observation to define the set of node-object triplets that would be beneficial for swapping at placement $\mathcal{A}$,

$$\begin{aligned}T(\mathcal{A})=&\{(i,o,p)|i\in\mathcal{N}, o\in\mathcal{A}_{\mathcal{P}(i)}\setminus\mathcal{R}_i, p\in\mathcal{A}_i,\\ &\big((p\in\mathcal{R}_i\setminus\mathcal{A}_i)\vee(p\notin\mathcal{R}_i\setminus\mathcal{A}_i\wedge I(i,o,p)=1)\big)\}.\end{aligned}$$

The algorithm terminates at iteration k if the set $T(\mathcal{A}(k))$ is empty. The pseudo-code of LGS is shown in Fig. 3.

To complete the definition of the algorithm, we now describe a greedy algorithm to choose the set $S(\mathcal{A}(k)) \subseteq T(\mathcal{A}(k))$ at iteration k. Given $T(\mathcal{A}(k))$, we choose a node i_k with a child that would like to swap (i.e., $\exists j \in \mathcal{C}(i_k)$ and $(j, o, p) \in T(\mathcal{A}(k))$). Given i_k we select the best swap (j_k, o_k, p_k) of its children, i.e., the one that maximizes the local cost saving in the subtree $\mathcal{N}_{i_k}$ (swap with parent), and we then allow every child node $j \in \mathcal{C}(i_k)$ to insert into its cache objects $o \in \mathcal{A}_{i_k}(k) \cup \{p_k\}$, if doing so would increase the local cost saving (copy from parent). The algorithm is shown in Algorithm 1.

In what follows we show that even though LGS is based on local information only, the swaps in fact decrease the global cost.

LGS Algorithm

```
1: $k \leftarrow 0$
2: while $|T(\mathcal{A}(k))| > 0$ do
3:     $\mathcal{A}(k+1) \leftarrow \mathcal{A}(k)$
4:     for each $(i,o,p) \in S(\mathcal{A}(k))$ do
5:         $\mathcal{A}_i(k+1) \leftarrow (\mathcal{A}_i(k) \cup \{o\} \setminus \{p\})$
6:         if $p \notin \mathcal{A}_{\mathcal{P}(i)}(k)$ then
7:             $\mathcal{A}_{\mathcal{P}(i)}(k+1) \leftarrow (\mathcal{A}_{\mathcal{P}(i)}(k) \cup \{p\} \setminus \{o\})$
8:         end if
9:     end for
10:    $k \leftarrow k+1$
11: end while
```

Figure 3: Pseudo-code of the *LGS* algorithm

Algorithm 1 $S(\mathcal{A}(k)) = \text{populateS}(\mathcal{A}(k), i_k)$

```
1: Select the best swapping opportunity at the children of $i_k$,
       $$(j_k, o_k, p_k) \leftarrow \arg\max_{\{(j,o,p)\in T(\mathcal{A}(k)) | j \in \mathcal{C}(i_k)\}} \sum_{n\in\mathcal{N}_j} c_{i,j}(w_n^o - w_n^p)$$
   $S(\mathcal{A}(k)) \leftarrow (j_k, o_k, p_k)$
2: Further decrease the cost function through allowing nodes in $\mathcal{C}(i_k)$ to insert
   objects available at $\{\mathcal{A}_{i_k}(k) \cup \{p_k\}\}$.
   $PE_j \leftarrow (\mathcal{A}_{i_k}(k) \cup \{p_k\}) \cap \mathcal{A}_j(k)$
   $PO_j \leftarrow (\mathcal{A}_{i_k}(k) \cup \{p_k\}) \setminus \mathcal{R}_j(k)$
   while $\exists(j,o,p)$ s.t. $o \in PO_j$ and $p \in PE_j$ and
       $$(p \in \mathcal{R}_j \setminus \mathcal{A}_j(k)) \vee (p \notin \{\mathcal{R}_j \setminus \mathcal{A}_j(k)\} \wedge I(j,o,p) = 1)) \qquad (6)$$
   do
       $S(\mathcal{A}(k)) \leftarrow S(\mathcal{A}(k)) \cup \{(j,o,p)\}$
       $PE_j \leftarrow PE_j \setminus \{p\}$
       $PO_j \leftarrow PO_j \setminus \{o\}$
   end while
```

Lemma 2. *The global cost $C(\mathcal{A})$ decreases strictly upon every swap.*

Proof. Consider $(i,o,p) \in S(\mathcal{A}(k))$ at iteration k. For every node $j \in \mathcal{N} \setminus \mathcal{N}_i$ it holds $d_{j,i} = d_{j,\mathcal{P}(i)} + c_{i,\mathcal{P}(i)} = d_{j,\mathcal{P}(i)}$, hence $d_j(o,\mathcal{A}(k+1)) = d_j(o,\mathcal{A}(k))$ and $d_j(p,\mathcal{A}(k+1)) = d_j(p,\mathcal{A}(k))$. Consequently, $C_j(\mathcal{A}(k+1)) = C_j(\mathcal{A}(k))$ for all $j \in \mathcal{N} \setminus \mathcal{N}_i$.
Consider now node $j \in \mathcal{N}_i$. Since $S(\mathcal{A}(k)) \subseteq T(\mathcal{A}(k))$, it follows that $o \notin \mathcal{R}_i(k)$ and $o \in \mathcal{A}_{\mathcal{P}(i)}(k)$. Hence $d_j(o,\mathcal{A}(k)) = d_{j,i} + c_{\mathcal{P}(i),i}$, $d_j(o,\mathcal{A}(k+1)) = d_{j,i}$, and the

difference in the cost $\Delta C(k+1)$ before and after the swap is

$$\begin{aligned}\Delta C(k+1) &= \sum_{j\in\mathcal{N}_i} [C_j(\mathcal{A}(k+1)) - C_j(\mathcal{A}(k))] \\ &= \sum_{j\in\mathcal{N}_i} \left[w_j^o d_{j,i} - w_j^o(d_{j,i} + c_{\mathcal{P}(i),i}) + w_j^p d_j(p,\mathcal{A}(k+1)) - w_j^p d_j(p,\mathcal{A}(k))\right] \\ &= \sum_{j\in\mathcal{N}_i} \left[-w_j^o c_{\mathcal{P}(i),i} + w_j^p \left(d_j(p,\mathcal{A}(k+1)) \; - d_j(p,\mathcal{A}(k))\right)\right].\end{aligned}$$

Similarly, $S(\mathcal{A}(k)) \subseteq T(\mathcal{A}(k))$ implies that $p \in \mathcal{A}_i(k)$, hence $d_j(p,\mathcal{A}(k)) \le d_{j,i}$. We now distinguish between two cases. If $d_j(p,\mathcal{A}(k)) < d_{j,i}$, then $d_j(p,\mathcal{A}(k+1)) = d_j(p,\mathcal{A}(k))$, which implies that $\Delta C(k+1) < 0$. Otherwise, if $d_j(p,\mathcal{A}(k)) = d_{j,i}$, then $d_j(p,\mathcal{A}(k+1)) = d_{j,i} + c_{\mathcal{P}(i),i}$. Since $I(i,o,p)=1$, then $\Delta C(k+1) = c_{\mathcal{P}(i),i}(\sum_{j\in\mathcal{N}_i}(w_j^p - w_j^o)) < 0$. This proves the lemma. □

We can use this result to show that the algorithm terminates after a finite number of iterations.

Theorem 3. *The* LGS *algorithm terminates after a finite number of iterations.*

Proof. Consider iteration k of the *LGS* algorithm. Call $s(\mathcal{A})$ the object placement that results from applying swap $s = (j,o,p)$ to placement $\mathcal{A}$. It follows from the proof of Lemma 2 that for any swap $s = (j,o,p) \in S(\mathcal{A}(k))$ and every node $l \in \mathcal{N} \setminus \mathcal{N}_j$, it holds $\{\mathcal{R}_j(\mathcal{A}(k)) \cup \mathcal{A}_{i_k}(k)\} = \{\mathcal{R}_j(s(\mathcal{A}(k))) \cup s(\mathcal{A}_{i_k}(k))\}$ and hence $C_l(s(\mathcal{A}(k))) = C_l(\mathcal{A}(k))$. Since for every $j,l \in \mathcal{C}(i_k), j \neq l$ it holds $l \notin \mathcal{N}_j$, we can consider each node $j \in \mathcal{C}(i_k)$ separately.
Consider swap $s = (j,o,p) \in S(\mathcal{A}(k))$. It follows from (6) that either $p \in \mathcal{R}_j \setminus \mathcal{A}_j(k)$ or $I(j,o,p) = 1$. Therefore, from the proof of Lemma 2, it follows that $C_l(s(\mathcal{A}(k))) \le C_l(\mathcal{A}(k))$ for all $l \in \mathcal{N}_j$. In particular, for swap $s_k = (j_k,o_k,p_k) \in T(\mathcal{A}(k))$, it holds that $I(j_k,o_k,p_k) = 1$, which implies $C_{j_k}(s_k(\mathcal{A}(k))) < C_{j_k}(\mathcal{A}(k))$. Since $\times_{i\in\mathcal{N}}\mathfrak{A}_i$ is a finite set, $C(\mathcal{A}(k))$ can not decrease indefinitely and the *LGS* algorithm terminates after a finite number of iterations. □

Besides guaranteed to converge starting from an arbitrary initial placement, a nice property of *LGS* is that if started from an optimal placement, the algorithm is stable in the sense that it does not make any changes, as we show next.

Corollary 1. *An optimal content placement $\bar{A}$ is stable under the* LGS *algorithm.*

Proof. From Lemma 2 and Theorem 3 it follows that $C(\mathcal{A}(k+1)) < C(\mathcal{A}(k))$ for any swap $s = (j,o,p) \in S(\mathcal{A}(k))$. By definition $\nexists \mathcal{A}' \in \times_{i\in\mathcal{N}}\mathfrak{A}_i$ s.t. $C(\mathcal{A}') < C(\bar{\mathcal{A}})$, hence the result. □

For simplicity, we restricted ourselves to a single i_k per iteration when defining $S(\mathcal{A}(k))$, but the above results hold for any set of nodes that are not each others' descendants, hence the algorithm can be executed in parallel.

4.2 *h*-Push Down Algorithm

In the *LGS* algorithm, every node i swaps objects based on the information about the object placement and the aggregate demand for objects at its descendants $\mathcal{D}(i)$. In the following we provide a distributed algorithm that allows node i to leverage additional information on placements and on aggregate demands for objects. In the *h-Push Down* algorithm, every node i has information about the placement $\mathcal{A}_{\mathcal{N}_j}$ and about the object demands w_k^o, $k \in \mathcal{N}_j$, for every ancestor j that lies within its information horizon h, i.e., for $j = \mathcal{P}^l(i)$ for $0 \leq l \leq h$.

The algorithm starts with an object placement $(\mathcal{A}_i(0))_{i \in \mathcal{N}}$ in which each node $i \in \mathcal{N}$ stores K_i objects that have the highest aggregated demands in the subnetwork $\mathcal{N}_i$ and that are not available in the cache of node i's descendants $\mathcal{D}(i)$. An iteration of the algorithm consists of two moves. The first move is an eviction operation at some node i. This is followed by a *push-down* move, which is a chain of evictions and insertions performed by the ancestors of node i.

Central to the algorithm is the LCA of node i and the node from which node i would retrieve object o in the placement $(\varnothing, \mathcal{A}_{-i})$, i.e., if it had no objects cached,

$$P_i^o(\mathcal{A}_{-i}) \triangleq \mathrm{LCA}\left(i, \underset{\{j \in \mathcal{N} \setminus \{i\} | o \in \mathcal{A}_j\}}{\arg\min} d_{i,j}\right). \tag{7}$$

Similarly, we define $P_i^o(\mathcal{A})$ for placement $\mathcal{A}$, i.e., $P_i^o(\mathcal{A}) = i$ if $o \in \mathcal{A}_i$, otherwise $P_i^o(\mathcal{A}) = P_i^o(\mathcal{A}_{-i})$. In the *h-Push Down* algorithm, a node i can only initiate a move, and therefore evict one object o, if o is cached at node i's descendants or if $P_i^o(\mathcal{A}_{-i})$ lies within node i's information horizon, i.e., $P_i^o(\mathcal{A}_{-i}) = \mathcal{P}^l(i)$ for some $0 < l \leq h$. We use $\mathrm{Z}_i(\mathcal{A})$ to denote the set of objects that are candidate for eviction at node i under placement $\mathcal{A}$, i.e.,

$$\mathrm{Z}_i(\mathcal{A}) = \{o \in \mathcal{A}_i | P_i^o(-\mathcal{A}) \in \bigcup_{l=0}^{h} \mathcal{P}^l(i) \vee o \in \bigcup_{j \in \mathcal{D}(i)} \mathcal{A}_j\}.$$

We use $\Delta C_{\mathrm{EV}}(i, o, \mathcal{A}) \triangleq C(\mathcal{A}) - C(\mathcal{A}_i \setminus \{o\}, \mathcal{A}_{-i})$ to denote the change in the global cost caused by the eviction of object o at node i. Observe that $\Delta C_{\mathrm{EV}}(i, o, \mathcal{A}) \leq 0$.

Figure 4 shows the pseudo-code of the push down move of the *h-Push Down* algorithm. The following lemma shows an important property of the push down move.

Lemma 3. *A push-down move* $\mathcal{A}' = \mathrm{PushDown}(i, \mathcal{A})$ *always decreases the global cost by*

$$\Delta C_{\mathrm{PD}}(i, \mathcal{A}) \triangleq C(\mathcal{A}) - C(\mathcal{A}') = \sum_{t=0}^{l(i)} c_{\mathcal{P}^{t+1}(i), \mathcal{P}^t(i)} \sum_{j \in T(t)} w_j^{o^t},$$

where $T(t) = \{j \in \mathcal{N}_{\mathcal{P}^t(i)} | P_j^{o^t}(\mathcal{A}) = \mathcal{P}^{t+1}(i)\}$.

$\mathcal{A}' = \text{PushDown}(i, \mathcal{A})$

```
1: $t \leftarrow 0$
2: $\mathcal{A}^0 \leftarrow \mathcal{A}$
3: do
4:     $n \leftarrow \mathcal{P}^t(i)$
5:     $o^t \leftarrow \arg\min_{o \in \mathcal{A}^t_{\mathcal{P}(n)}} C(\mathcal{A}^t_n \cup \{o\}, \mathcal{A}^t_{-n})$
6:     $\mathcal{A}^{t+1}_n \leftarrow \mathcal{A}^t_n \cup \{o^t\}$
7:     $\mathcal{A}^{t+1}_{\mathcal{P}(n)} \leftarrow \mathcal{A}^t_{\mathcal{P}(n)} \setminus \{o^t\}$
8:     $t \leftarrow t + 1$
9: while $n \neq n_0$
10: return $\mathcal{A}'$
```

Figure 4: Pseudo-code of the *push-down* move of the *h-Push Down* algorithm.

Proof. Consider iteration t of the push-down move $\mathcal{A}' = \text{PushDown}(i, \mathcal{A})$. Since $c_{n,\mathcal{P}(n)} = 0$, for all $j \in \mathcal{N} \setminus \mathcal{N}_n$ it holds $d_j(o^t, \mathcal{A}^t) = d_j(o^t, \mathcal{A}^{t+1})$. Furthermore, for all $j \in \mathcal{N}_n$ we need to distinguish between two cases. If $P^{o^t}_j(\mathcal{A}^t) \neq \mathcal{P}(n)$, then $P^{o^{t+1}}_j(\mathcal{A}) = P^{o^t}_j(\mathcal{A}^t)$ and $d_j(o^t, \mathcal{A}^t) = d_j(o^t, \mathcal{A}^{t+1})$. It follows that, if $j \notin T(t)$, then $C^o_j(\mathcal{A}^t) - C^o_j(\mathcal{A}^{t+1}) = 0$. Otherwise, $P^{o^t}_j(\mathcal{A}^t) = \mathcal{P}(n)$ implies $P^{o^t}_j(\mathcal{A}^{t+1}) = n$, and hence $C^o_j(\mathcal{A}^t) - C^o_j(\mathcal{A}^{t+1}) = w^{o^t}_j c_{\mathcal{P}(n),n}$. By summing over all the $l(i)$ iterations of the push-down move, we prove the lemma. □

The pseudo-code of the *h-Push Down* algorithm is shown in Figure 5. We start with showing that the algorithm terminates in a finite number of iterations.

Theorem 4. *The h-Push Down algorithm terminates after a finite number of iterations.*

Proof. We prove the theorem by showing that the global cost $C(\mathcal{A})$ decreases at every iteration of the h-Push Down algorithm. From Lemma 3 it follows that

$$\Delta C_{\text{PD}}(i_k, (\mathcal{A}(k)_{i_k} \setminus \{o^k\}, \mathcal{A}(k)_{-i_k})) \geq \Delta C^h_{\text{PD}}(i_k, \mathcal{A}(k)). \quad (8)$$

By definition, the variation of the global cost at iteration k can be written as the sum of the variation due to the eviction and the variation due to push down move, i.e., $\Delta C_{\text{EV}}(i_k, o^k, \mathcal{A}(k)) + \Delta C_{\text{PD}}(i_k, (\mathcal{A}(k)_{i_k} \setminus \{o^k\}, \mathcal{A}(k)_{-i_k})) = C(\mathcal{A}(k)) - C(\mathcal{A}(k+1))$. The proof of the theorem follows from (8). □

Furthermore, similar to *LGS*, the algorithm does not make any changes to an optimal placement, as shown next.

Corollary 2. *The optimal content placement $\bar{A}$ is stable with respect to the h-Push Down algorithm.*

h-Push Down Algorithm

```
1: $k \leftarrow 0$
2: $\mathrm{Z}^0 \leftarrow \{i \in \mathcal{N} \text{ such that } |\, \mathrm{Z}_i(\mathcal{A}(0))| > 0\}$
3: $\mathcal{A} \leftarrow \mathcal{A}(0)$
4: while $|\, \mathrm{Z}^k \,| > 0$ do
5:     Pick $i_k \in \mathrm{Z}^k$
6:     Compute the least cost eviction $o^k \leftarrow \arg\min_{o \in \mathrm{Z}_{i_k}} |\Delta C_{\mathrm{EV}}(i_k, o, \mathcal{A})|$
7:     Compute $\Delta C^h_{\mathrm{PD}}(i_k, \mathcal{A})$ as
           $$\Delta C^h_{\mathrm{PD}}(i_k, \mathcal{A}) = \sum_{t=0}^{\min(h, l(i_k))} c_{\mathcal{P}^{t+1}(i_k), \mathcal{P}^t(i_k)} \sum_{j \in T(t)} w_j^{o^t},$$
8:     if $\Delta C^h_{\mathrm{PD}}(i_k, \mathcal{A}) + \Delta C_{\mathrm{EV}}(i_k, o^k, \mathcal{A}) > 0$ then
9:         $\mathcal{A}(k+1) \leftarrow \mathrm{PushDown}(i, (\mathcal{A}_{i_k} \setminus \{o^k\}, \mathcal{A}_{-i_k}))$
10:        $k \leftarrow k + 1$
11:        $\mathcal{A} \leftarrow \mathcal{A}(k)$
12:        $\mathrm{Z}^k \leftarrow \{i \in \mathcal{N} \text{ such that } |\, \mathrm{Z}_i(\mathcal{A}(k))| > 0\}$
13:    else
14:        $\mathrm{Z}^k \leftarrow \mathrm{Z}^k \setminus \{i_k\}$
15:    end if
16: end while
```

Figure 5: Pseudo code of the *h-Push Down* algorithm.

Proof. The proof is analogous to the proof of Corollary 1. □

Observe that the computation of $\Delta C^h_{\mathrm{PD}}(i_k, \mathcal{A}(k))$ depends only on the object demands and the placements at the nodes in the set $\mathcal{N}_{\mathcal{P}^h(i_k)}$. Furthermore, in order to compute $\Delta C_{\mathrm{EV}}(i_k, o^k, \mathcal{A}(k))$, node i_k only requires information about placements and demands in the subnetwork $\mathcal{N}_{P_i^{o^k}(\mathcal{A}_{-i}(k))}$, which lies within node i_k's information horizon h.

5 Numerical Results

We use simulations to evaluate the approximation ratio and the convergence rate of the proposed algorithms. To generate backhaul topologies, we use the *Manhattan* model, in which $|\mathcal{N}|$ nodes are randomly placed on an $|\mathcal{N}| \times |\mathcal{N}|$ grid. Given the node placement, we build a weighted complete graph by setting the weight on edge (i, j) equal to the Euclidean distance between nodes i and j, computed based on their coordinates. We then run *Kruskal*'s algorithm [12] on the resulting weighted complete graph to compute a minimum spanning tree to obtain the topology $\mathcal{G}$. We consider two different cost models. In the *distance* cost model the edge costs $c_{\mathcal{P}(i),i}$ are equal to the weights used for generating the tree. In the *descendants* cost model

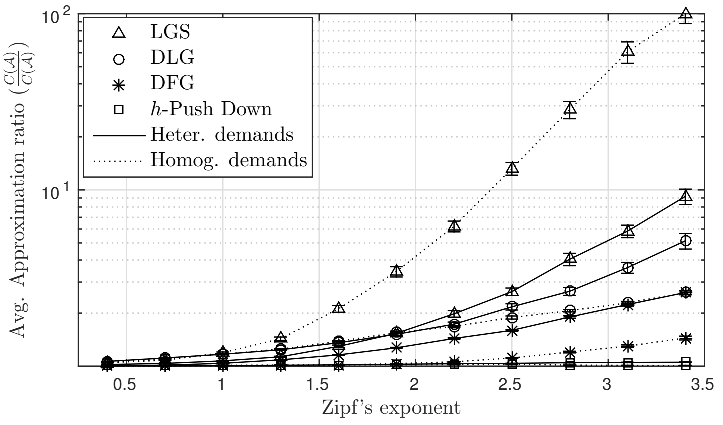


Figure 6: Average approximation ratio vs. Zipf exponent for the *LGS*, *DLG*, *DFG*, and *h-Push Down* algorithms. *Heterogeneous* and *homogeneous demands*, $|\mathcal{O}| = 100$, $|\mathcal{N}| = 20$, $K_i = 2$.

the edge costs $c_{\mathcal{P}(i),i}$ are proportional to the size of the subnetwork $\mathcal{N}_i$, in which case the cost of using a link is proportionate to the size of the subtree that the link is serving.

We consider two models for the object demands w_i^o at the nodes. In the case of *homogeneous demands*, the object demands have the same rank at all nodes. In the case of *heterogeneous demands*, every demand w_i^o for object o at node i is ranked as in the case of *homogeneous demands* with 0.5 probability. With 0.5 probability, the rank of w_i^o is picked uniformly at random. The results shown are the averages of 500 simulations, and the error bars show 95% confidence intervals.

As a baseline for comparison, we use a selfish distributed algorithm called *Distributed Local-Greedy* (DLG), which is based on global information about the object demands and placements at every node of the network. Following the *DLG* algorithm, starting from a randomly chosen alloction, at iteration k node i_k optimizes its placement of objects $\mathcal{A}_{i_k}(k)$ so as to minimize the cost for serving the requests from the local cell site, given the placement of objects $\mathcal{A}_{-i_k}(k)$ at the other nodes in the network [13, 14, 15]. As there is no guarantee that the *DLG* algorithm terminates [15], we run it for $|\mathcal{N}|$ iterations and we set $i_k = k$. Note that although *DLG* is seemingly similar to *DFG*, *DFG* minimizes the global cost based on global information, while *DLG* minimizes the local cost based on global information, hence it is algorithmically simpler.

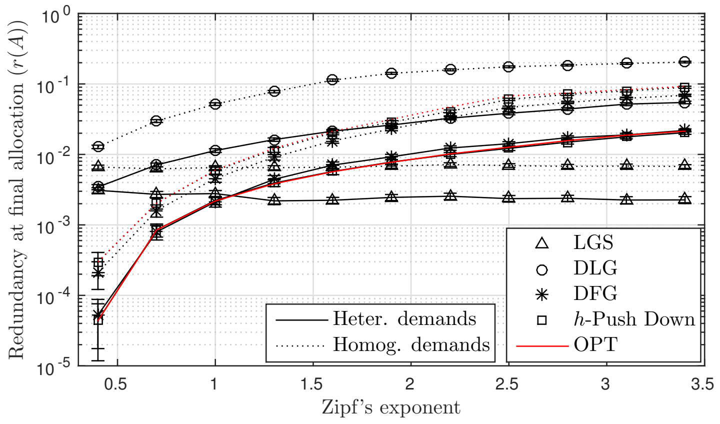


Figure 7: Redundancy $r(\mathcal{A})$ vs. Zipf exponent for *LGS*, *DLG*, *DFG*, and *h-Push Down* and for the optimal placement. *Heterogeneous* and *homogeneous demands*, $|\mathcal{O}| = 100$, $|\mathcal{N}| = 20$, $K_i = 2$.

5.1 Performance of distributed algorithms

In order to compare the performance of the proposed algorithms, as well as to evaluate the tightness of the analytical results, we computed the optimal placement $\bar{\mathcal{A}}$ and the cost-approximation ratio $C(\mathcal{A})/C(\bar{\mathcal{A}})$ for each algorithm. To make the computation of the optimal placement feasible, we considered a relatively small scenario with $|\mathcal{N}| = 20$, $|\mathcal{O}| = 100$ and $K_i = 2$ for all $i \in \mathcal{N}$. Figure 6 shows the cost-approximation ratio as a function of the Zipf exponent of the object demand distribution for *LGS*, *DLG*, *DFG* and for the *h-Push Down* algorithm with global information, i.e., for $h = \max_{i \in \mathcal{N}} l(i)$, for the *descendants* cost model.

The most salient feature of the figure is that the approximation ratio of the *LGS* algorithm increases exponentially with the Zipf exponent at a fairly high rate. The reason for the poor performance in the case of homogeneous demands is that the *LGS* algorithm populates the set $S(\mathcal{A}(k))$ exclusively based on the rankings of the object demands and not based on their values. As the Zipf exponent increases, the demand of the most popular content increases and the optimal solution might differ significantly from the allocation reached by the *LGS* algorithm. In order to validate this hypothesis, we computed the redundancy of a placement $\mathcal{A}$ using the index

$$r(\mathcal{A}) = \frac{\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N} \setminus \{i\}} \left(1 - \frac{\min(K_i, K_j) - |\mathcal{A}_i \cap \mathcal{A}_j|}{\min(K_i, K_j)}\right)}{|\mathcal{N}|(|\mathcal{N}| - 1)}. \tag{9}$$

Intuitively, $r(\mathcal{A})$ is the average ratio of common objects present between all pairs of placements $\mathcal{A}_i$ and $\mathcal{A}_j$. In Figure 7 we plot the average $r(\mathcal{A})$ index of the final placements reached by the algorithms, for the same scenario as Figure 6. The figure

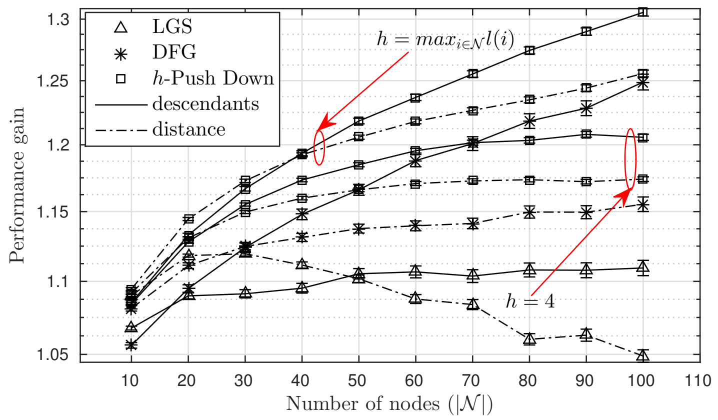


Figure 8: Performance gain vs number of nodes $|\mathcal{N}|$ for the *h-Push Down*, *LGS* and *DFG* algorithms on the Manhattan graph with *descendants* and *distance* cost model, $|\mathcal{O}| = 5000$, $K_i = 20$.

confirms that as the Zipf exponent increases, the *LGS* algorithm fails to introduce enough redundancy, which explains its poor performance.

Comparing the performance of *h-Push Down* to that of *DFG* we observe that *h-Push Down* (with global information) performs better than *DFG*, which is also reflected by the redundancy index, which is very close to the optimal (cf. Fig. 6). Finally, it is noteworthy that the *DLG* algorithm, which corresponds to selfish local optimization, fails to achieve performance close to the optimal, despite the availability of global information.

In order to evaluate the performance of the algorithms for larger scenarios, in the following we use the *DLG* algorithm as a baseline for comparison, as it is prohibitive to compute the optimal placement. Recall that the *DLG* algorithm optimizes the placement of objects in order to minimize the local cost, which would make it a reasonable simple choice in absence of more elaborate distributed algorithms.

To capture the performance of the algorithms relative to *DLG* we define the performance gain of an algorithm as the ratio between the cost of the placement reached by the *DLG* algorithm and the cost of the placement reached by the algorithm. Figure 8 shows the performance gain for the *LGS*, *DFG* and *h-Push Down* (for two values of the information horizon h) algorithms, as a function of the number of nodes for $K_i = 20$. The results are shown for *heterogeneous demands* using a Zipf exponent of 1, for the two cost models. We observe that the performance gain for the *DFG* and the *h-Push Down* algorithms increases with the number of nodes. Furthermore, the figure shows that *h-Push Down* outperforms *DFG* (i.e., it is close to optimal) for both values of the horizon h. The figure also shows that *LGS* performs just slightly better than *DLG*, with a decreasing gain as the network

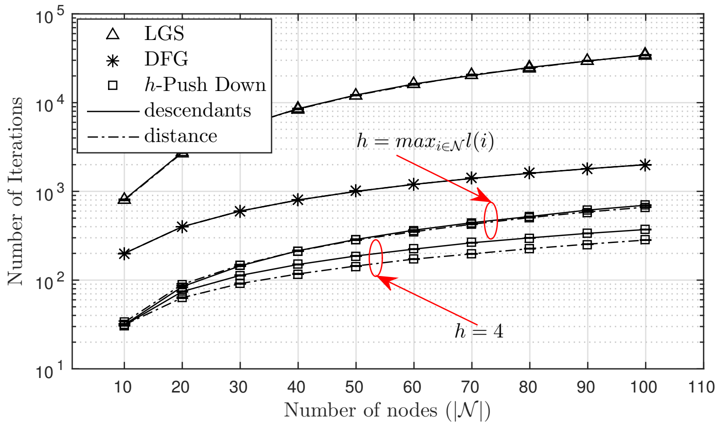


Figure 9: Number of iterations vs number of nodes $|\mathcal{N}|$ for the *h-Push Down*, *LGS* and *DFG* algorithms on the Manhattan graph with *descendants* and *distance* cost model, $|\mathcal{O}| = 5000$, $K_i = 20$.

size increases.

Figure 9 shows the number of iterations needed to compute the final object placement corresponding to the results showen in Figure 8. Recall that the *DFG* algorithm starts with an empty allocation and terminates in $\sum_{i \in \mathcal{N}} K_i$ iterations, and can thus be used a baseline in terms of convergence. The results show that *LGS* performs worst, while *h-Push Down* for $h = 4$ requires almost an order of magnitude less iterations to terminate than *DFG*.

Figure 10 shows the performance gain as a function of the cache sizes for $|\mathcal{N}| = 50$. The figure shows that for higher cache sizes the performance gain of the *DFG* and *h-Push Down* algorithms over the *DLG* algorithm increases faster than exponentially. In the case of global information, the *h-Push Down* algorithm outperforms the *DFG* algorithm, while in the case of non-global information, i.e., for $h = 4$, it achieves performance close to the *DFG* algorithm. Furthermore, the performance gap between the *h-Push Down* algorithm with global and non-global information increases for higher cache sizes. The figure also confirms that the *LGS* and *DLG* algorithms achieve a comparable total cost.

5.2 Impact of the information horizon (h)

Finally, we evaluate the impact of the information horizon h on the performance of h-Push Down. We define the performance gain $\mathrm{PG}^h(\mathcal{A})$ for horizon h as the ratio between the cost of the placement $\mathcal{A}^1$ reached by the h-Push Down algorithm with $h = 1$ and the cost of the placement $\mathcal{A}^h$ reached with horizon h, i.e. $\mathrm{PG}^h(\mathcal{A}) = \frac{C(\mathcal{A}^1)}{C(\mathcal{A}^h)}$.

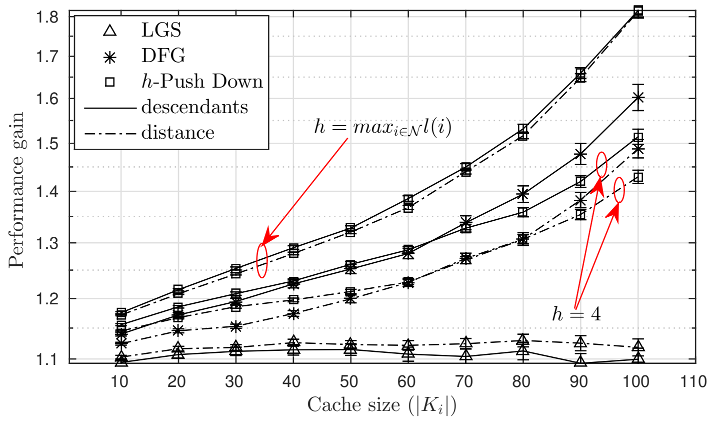


Figure 10: Performance gain vs. cache size K_i for *h-Push Down*, *LGS* and *DFG* on the Manhattan graph with *descendants* and *distance* cost model. Results for $|\mathcal{O}| = 5000$, $|\mathcal{N}| = 50$.

Figures 11 and 12 show the performance gain $\text{PG}^h(\mathcal{A})$ and the number of iterations, respectively, for the *h-Push Down* algorithm as a function of the information horizon h for $|\mathcal{N}| = 100$ and two different cache sizes K_i. We plot the performance gain $\text{PG}^h(\mathcal{A})$ for the same cost and object demands models as in Figures 8 and 10. We observe that the performance gain increases with a decreasing marginal gain in h, making the algorithm perform fairly well with limited available information (low h). Furthermore, the same observation holds for the convergence time, hence a moderate value of h provides a good trade-off between performance and convergence time. Figure 11 also shows that as the horizon h increases, the performance gain increases more in the case of the *descendants* cost model than in the case of the *distance* cost model. The reason is that as the horizon h increases, the nodes have access to the cost of edges between nodes at lower levels of the tree (i.e., closer to the root), which in the case of the *descendants* cost model are the edges with highest cost, and thus they have a higher impact on the total cost.

6 Related Work

Closest to ours are recent works on content placement in networks [16, 17, 5, 7]. The authors in [16] provide an algorithm for computing the optimal placement in a hierarchical network by reducing the content placement problem to a minimum-cost flow problem. Motivated by the computational complexity of the problem, they design, and further improve in [17] a distributed amortizing algorithm that achieves a constant factor approximation. The model considered in [16, 17] is based on the ultrametric cost model introduced in [6], which differs from our model on the

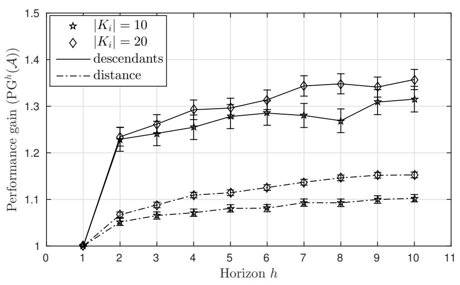


Figure 11: Performance gain vs. horizon h for cache sizes $K_i \in \{10, 20\}$ on the Manhattan graph with *descendants* and *distance* cost models. Results for $|\mathcal{O}| = 5000$ and $|\mathcal{N}| = 100$.

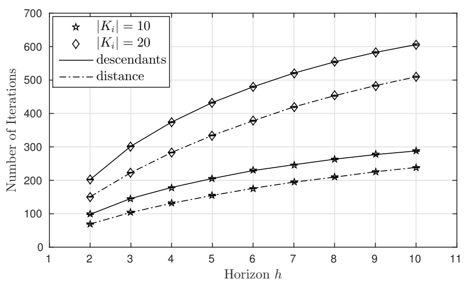


Figure 12: Number of iterations vs horizon h for two values of cache sizes $K_i \in \{10, 20\}$ on the Manhattan graph with *descendants* and *distance* cost models. Results for $|\mathcal{O}| = 5000$ and $|\mathcal{N}| = 100$.

assumption of symmetric costs between nodes. The authors in [5] give insights in the structure of the optimal placement in a regular two level hierarchical network, and they develop a greedy distributed 2-approximation algorithm. The authors in [11] consider a hybrid network with in-network caching and they propose a $(1 - 1/e)$-approximation greedy algorithm. A more generic cost model was considered in [7], where the authors only assume that the nodes are located in a common metric space. They develop a 10-approximation algorithm by rounding the optimal solution of the LP-relaxation of the problem, but they do not consider distributed algorithms.

Related to ours are recent works on game theoretical analyses of distributed selfish replication on graphs [13, 18, 19, 20, 21, 14, 15], as they can serve as a basis for distributed content placemenet algorithms. Equilibrium existence when the access costs are homogeneous and nodes form a complete graph were provided in [13], and results on the approximation ratio (referred to as the price of anarchy) were provided in [18, 19] for homogeneous costs and a complete graph. Non-complete graphs were considered in [20, 21, 14], and results on the approximation ratio of a distributed greedy algorithm were given for the case of unit storage capacity and an infinite number of objects in [20]. [21] considered a variant of the problem where nodes can replicate a fraction of objects, and showed the existence of equilibria, while convergence results were provided for the integer problem in [14] in the case of homogeneous neighbor costs. The case of heterogeneous neighbor costs, for which the non-convergence of distributed greedy replication was shown in [15] is a generalization of our model, and thus the negative result provided in [15] may not apply to our case. Different from these works, in this paper we consider caches managed by a single entity, and thus we consider the minimization of the total cost as opposed to the selfish minimization of the cost of the individual nodes.

7 Conclusion

We considered the problem of minimizing the bandwidth demand in a mobile backhaul through cooperative caching, and formulated it as a 0-1 integer linear program. We proposed a 2-approximation distributed algorithm that is based on global information. Furthermore, we proposed a low complexity distributed algorithm based on local information, and an algorithm with an adjustable level of available information. We proved convergence and stability of the algorithms. We used extensive simulations to evaluate the performance of the proposed algorithms. Our results show that local information is insufficient for good cooperative caching performance, but the proposed h-Push Down algorithm achieves consistently good performance despite limited information availability, consistently better than greedy optimization based on global information.

References

[1] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Scalable Mobile Backhauling via Information-Centric Networking," in *Proc. of IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*, 2015, pp. 1–6.

[2] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, "Understanding traffic dynamics in cellular data networks," in *Proc. of IEEE INFOCOM*, 2011, pp. 882–890.

[3] H. Pinto, J. M. Almeida, and M. A. Gonçalves, "Using early view patterns to predict the popularity of youtube videos," in *Proc. of ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, 2013, pp. 365–374.

[4] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, "A survey on predicting the popularity of web content," *Journal of Internet Services and Applications*, vol. 5, no. 1, p. 8, 2014.

[5] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," in *Proc. of IEEE INFOCOM*, 2010.

[6] D. Karger, T. Leightonl, D. Lewinl, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web," *in Proc. of ACM STOC*, pp. 654–663, 1997.

[7] I. D. Baev and R. Rajaraman, "Approximation algorithms for data placement in arbitrary networks," in *Proc. of ACM SODA*, 2001.

[8] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate Models for General Cache Networks," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.

[9] C. Fricker, P. Robert, and J. Roberts, “A versatile and accurate approximation for LRU cache performance,” in *Proc. of the 24th International Teletraffic Congress (ITC)*, 2012, pp. 1–8.

[10] Fisher, M. L., G. L. Nemhauser, and L. A. Wolsey, “ An analysis of approximations for maximizing submodular set functions - II,” *Mathematical Programming Study 8*, pp. 73–87, 1978.

[11] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, “On the complexity of optimal routing and content caching in heterogeneous networks,” in *Proc. of IEEE INFOCOM*, 2015, pp. 936–944.

[12] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–48, 1956.

[13] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, “Distributed selfish replication,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 12, pp. 1401–1413, 2006.

[14] V. Pacifici and G. Dán, “Convergence in Player-Specific Graphical Resource Allocation Games,” *IEEE J. Sel. Areas Commun.*, vol. 30, no. 11, pp. 2190–2199, 2012.

[15] ——, “Distributed Algorithms for Content Allocation in Interconnected Content Distribution Networks,” in *Proc. of IEEE INFOCOM*, 2015.

[16] M. R. Korupolu, C. Plaxton, and R. Rajaraman, “Placement Algorithms for Hierarchical Cooperative Caching,” *Journal of Algorithms*, vol. 38, no. 1, pp. 260–302, 2001.

[17] M. Korupolu and M. Dahlin, “Coordinated placement and replacement for large-scale distributed caches,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 6, pp. 1317–1329, 2002.

[18] G. Pollatos, O. Telelis, and V. Zissimopoulos, “On the social cost of distributed selfish content replication,” in *Proc. of IFIP/TC6 Networking*, 2008, pp. 195–206.

[19] E. Jaho, M. Karaliopoulos, and I. Stavrakakis, “Social similarity favors cooperation: the distributed content replication case,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 601–613, 2013.

[20] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz, “Selfish caching in distributed systems: a game-theoretic analysis,” in *Proc. of ACM PODC*, 2004, pp. 21–30.

[21] G. Dán, “Cache-to-cache: Could ISPs cooperate to decrease peer-to-peer content distribution costs?” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1469–1482, 2011.

Paper **B**

Cache Bandwidth Allocation for P2P File Sharing Systems to Minimize Inter-ISP Traffic

Valentino Pacifici and Frank Lehrieder and György Dán

in IEEE/ACM Transactions on Networking (ToN), November 2014.

Cache Bandwidth Allocation for P2P File Sharing Systems to Minimize Inter-ISP Traffic

Valentino Pacifici*, Frank Lehrieder⋆, György Dán*

*School of Electrical Engineering,
KTH Royal Institute of Technology, Stockholm, Sweden

⋆Institute of Computer Science,
University of Würzbur, Würzburg, Germany

Abstract

Many Internet service providers (ISPs) have deployed peer-to-peer (P2P) caches in their networks in order to decrease costly inter-ISP traffic. A P2P cache stores parts of the most popular contents locally, and if possible serves the requests of local peers to decrease the inter-ISP traffic. Traditionally, P2P cache resource management focuses on managing the storage resource of the cache so as to maximize the inter-ISP traffic savings. In this paper we show that, when there are many overlays competing for the upload bandwidth of a P2P cache, then in order to maximize the inter-ISP traffic savings the cache's upload bandwidth should be actively allocated among the overlays. We formulate the problem of P2P cache bandwidth allocation as a Markov decision process, and propose three approximations to the optimal cache bandwidth allocation policy. We use extensive simulations and experiments to evaluate the performance of the proposed policies, and show that the bandwidth allocation policy that prioritizes swarms with a small ratio of local peers can improve the inter-ISP traffic savings in BitTorrent-like P2P systems by up to 30 to 60 percent.

1 Introduction

The number of peer-to-peer applications has increased significantly in recent years, and so has the amount of Internet traffic generated by peer-to-peer (P2P) applications. P2P traffic accounts for up to 70% of the total network traffic, depending on geographical location [1], and is a significant source of inter-ISP traffic. Inter-ISP traffic can be a source of revenue for tier-1 ISPs, but it is a source of transit traffic costs for ISPs at the lower levels of the ISP hierarchy, e.g., for tier-2 and tier-3 ISPs. Some ISPs have attempted to limit their costs due to P2P applications by throttling P2P traffic [2]. Nevertheless, the users of P2P applications constitute a significant share of the ISPs' customer base, and hence a solution that negatively affects the performance of P2P applications can result in a decrease of an ISP's revenues on the long term.

Recent research efforts have tried to decrease the amount of inter-ISP P2P traffic by introducing locality-awareness in the neighbor-selection policies of popular P2P applications, like BitTorrent [3–6]. Locality information can be provided by the ISPs [3–5] or can be obtained via measurements [6], and is used to prioritize nearby peers to distant ones when exchanging data. Through exchanging data primarily with nearby peers a P2P application can improve the locality of its traffic, and hence, can decrease inter-ISP traffic. Nevertheless, locality-aware neighbor selection can deteriorate the performance and the robustness of a P2P application [7].

To address the problem of increased inter-ISP traffic, many ISPs have deployed P2P caches [8, 9]. P2P caches, similar to web proxy caches, decrease the amount of inter-ISP traffic by storing the most popular contents in the ISP's own network, so that they do not have to be downloaded from peers in other ISPs' networks. According to measurement studies 30 to 80 percent of P2P traffic is cacheable [10, 11]. Nevertheless, the actual efficiency of a cache depends on two main factors. First, the amount of storage, which determines the share of the contents that can be kept in cache. Second, the available bandwidth of the cache, which determines the rate at which data can be served by the cache, if the data are in storage.

The goal of cache storage management is to maximize the probability that data are found in the cache when requested. The algorithms for cache storage management, called cache eviction policies, in the case of P2P caches differ significantly from those in the case of web proxy caching. Web objects are typically small, and consequently eviction policies can replace entire contents at once [12]. Objects in P2P systems are nevertheless typically too big to be replaced at once, so that eviction policies for P2P caches have to allow partial caching of contents [10,11]. By allowing partial caching, P2P eviction policies can achieve within 10 to 20 percent of the optimal offline eviction policy [10, 11].

The impact of the cache bandwidth and its management has received little attention, even though cache bandwidth can be costly, as caches are often priced based on their bandwidth [8, 9]. In the case of web proxy caching bandwidth management is not necessary, because the incoming inter-ISP traffic saving equals the amount of data served from the cache. In the case of a P2P cache the inter-ISP traffic saving is, however, not only determined by how much data the cache serves but also by the characteristics of the overlay to which the data is served [13].

The fundamental question we address in this paper is whether given a limited amount of P2P cache bandwidth, the bandwidth can be actively managed such as to minimize the amount of inter-ISP traffic. We make three important contributions to answer this question. First, we provide a mathematical formulation of the cache bandwidth allocation problem, and show the existence of a stationary optimal policy. Second, we use the proposed mathematical model and insights from [13, 14] to derive three allocation policies to approximate the optimal policy. Third, through simulations and through experiments on Planet-lab we show that by actively allocating the upload bandwidth between different overlays the inter-ISP traffic savings due to P2P caches can be improved significantly. We identify the heterogeneity of the ratio of local peers in the swarms as the key factor that determines the potential traffic savings.

The rest of the paper is organized as follows. In Section 2 we review the related work. In Section 3 we model the system and its evolution using a Markov jump process. In Section 4 we formulate the problem of cache bandwidth allocation and show the existence of an optimal cache bandwidth allocation policy. Section 5 describes three policies to approximate the optimal policy. In Section 6 we use simulation and experiment results to quantify the potential of the proposed bandwidth allocation policies and to provide insight into the characteristics of an optimal policy. Section 7 concludes the paper.

2 Related work

The solutions for ISP-friendly P2P application design proposed in the literature fall into three main categories: peer-driven, ISP-driven and caching [15]. Peer-driven solutions adapt the neighbor selection strategy of the peers by relying on measurements of latency [16], on autonomous system (AS) topology map information [5] or on third-party infrastructures like content delivery networks [6]. Motivated by the difficulty of inferring the ISPs' interests based on measurements [3, 4] investigated the use of ISP-provided information to influence peer selection. All these works make P2P systems more ISP-friendly by influencing the overlay construction, and are complementary to P2P caching.

Caching of P2P contents has been the subject of several works. Most works focused on the achievable cache hit ratios [17, 18], and on the efficiency of various cache eviction policies [10, 11]. Our work is orthogonal to the works on cache eviction policies, as we assume the existence of a cache eviction policy, and we consider the impact of allocating the cache's upload bandwidth between competing overlays on the amount of inter-ISP traffic generated by the overlays.

Cache upload bandwidth management for P2P video streaming systems was considered in [19, 20] in order to decrease the ISPs' incoming transit traffic. In the case of streaming the download rate of peers is determined by the video rate, and the received rate does not influence the peers' behavior. This makes the problem of cache bandwidth allocation for streaming systems significantly different from the problem considered in this paper. We do not only consider the impact of the cache upload rate on the instantaneous inter-ISP traffic, but also its impact on the system dynamic.

Close to our work is [21] where the authors studied the impact of different bandwidth reservation schemes between two overlays via simulations. They concluded that the impact of cache bandwidth allocation was minor, which can be attributed to the inefficiency of the cache bandwidth utilization under the considered schemes. Compared to [21] in this paper we give a mathematical formulation of the problem of cache bandwidth allocation, use analytical models of the swarm dynamics and the inter-ISP traffic to give insight into the characteristics of an optimal allocation policy, and use simulations and experiments to demonstrate the inter-ISP traffic savings achievable through cache bandwidth allocation.

In [22–24] the authors proposed schemes for bandwidth allocation among multiple swarms in P2P systems. In these works the initial seeder is the bandwidth allocator that attempts to maximize the total download rate of the system so as to minimize the download latencies of the peers. The most fundamental difference between our work and [22–24] is that we aim at minimizing the amount of inter-ISP traffic generated by the overlays. [22, 23] consider managed swarms, while in our work caching is performed transparently to the peers. In [23] the authors assume peers belonging to multiple swarms. The bandwidth allocation among swarms is implemented by these peers as they follow the prioritization scheme suggested by the coordinator. More similar to our work is [24], where the authors implement a simple model-based controller for server bandwidth in BitTorrent systems. Due to the large amount of data needed to parametrize the model, the authors question the practicality of their approach [24].

Our work relies on the analytical models of the system dynamics of BitTorrent-like systems in [13, 25–29]. These works used a Markovian model of the system dynamics of BitTorrent-like systems to model the service capacity and the scalability [25, 26], to evaluate the impact of peer upload rate allocation between two classes of peers [27], to assist the dimensioning of server assisted hybrid P2P content distribution [29], and to evaluate the impact of caches on the swarm dynamics and on the amount of inter-ISP traffic for a single overlay [13]. Our work differs significantly from these works, as we consider multiple overlays and use the fluid model of the system dynamics to get insight into the characteristics of an optimal P2P cache bandwidth allocation policy.

In our work we model the cache bandwidth allocation problem in P2P systems as a Markov Decision Process (MDP). MDPs were used in [30–32] to analyze schemes for incentivizing fair resource reciprocation and for discouraging free riding in P2P systems. Compared to [31–33], in our work we use a MDP to prove the existence of an optimal cache allocation policy.

3 System Model

In the following we describe our model of a multi-swarm file-sharing system and our model of cache bandwidth allocation. The model captures the effect of the cache bandwidth allocation on the evolution of the system.

We consider a set $\mathcal{I} = \{1, \dots, I\}$ of ISPs, and a set of swarms $\mathcal{S} = \{1, \dots, S\}$, whose peers are spread over the ISPs. Peers are either leechers, which download and upload simultaneously, or seeds, which upload only. Leechers arrive to swarm s according to a Poisson process with intensity λ_s, the arrival rate of leechers in ISP i is $\lambda_{i,s}$. The Poisson process can be a reasonable approximation of the arrival process over short periods of time [34], even if the arrival rate of peers varies over the lifetime of a swarm. We model the leechers' impatience by the abort rate θ. A leecher departs at this rate before downloading the entire content. Seeds depart from the swarm at rate γ, so that a seed stays on average $1/\gamma$ time in the swarm. The upload rate of peers is denoted by μ and their download rate by c. We focus on the case when $\mu < c$. For simplicity we consider that all files have the same size, and thus, μ and c can be normalized by the file size. Finally, we assume that leechers can use a share η of their upload rate due to partial content availability. This model of swarm dynamics was used in [13,25,26,28,29].

We denote by $X_{i,s}(t)$ the number of leechers in ISP i in swarm s at time t, and by $Y_{i,s}(t)$ the number of seeds in ISP i in swarm s at time t. $X_{i,s}(t)$ and $Y_{i,s}(t)$ take values in the countably infinite state space $\mathbb{N}_0$. As a shorthand we introduce $Z_{i,s}(t) = (X_{i,s}(t), Y_{i,s}(t))$ and $Z_s(t) = (Z_{i,s}(t))_{i\in\mathcal{I}}$. Finally, we denote the state of the swarms by $Z(t) = (Z_s(t))_{s\in\mathcal{S}}$.

Seeds and leechers in ISP i can upload and download data to and from peers in any ISP $j \in \mathcal{I}$. We define the publicly available upload rate $u^P_{i,s}(t)$ as the available upload rate located in ISP i that can be used by leechers of swarm s in any ISP. This quantity tantamounts the upload rate of the leechers and the seeds $u^P_{i,s}(t) = \mu(\eta X_{i,s}(t) + Y_{i,s}(t))$. A leecher cannot download from itself, therefore the publicly available upload rate in ISP i to a local leecher of swarm s is $u^{PL}_{i,s}(t) = \max[0, \mu(\eta(X_{i,s}(t) - 1) + Y_{i,s}(t)]$.

3.1 P2P Cache Bandwidth Allocation Policies

The ISPs, as they are located in the lower layers of the ISP hierarchy, are interested in decreasing the inter-ISP traffic generated by the peers. In order to decrease its inter-ISP traffic, ISP $i \in \mathcal{I}$ maintains a cache with upload *bandwidth capacity* $K_i < \infty$, which acts as an ISP managed super peer [8]. The abstraction of a P2P cache as a source of upload bandwidth is motivated by that P2P caches are often priced by their maximum upload rates. Since every ISP's goal is to decrease its own incoming inter-ISP traffic, it is reasonable to assume that the cache operated by ISP i only serves leechers in ISP i.

ISP i can implement an *active* cache bandwidth allocation policy to control the amount of cache bandwidth $\kappa_{i,s}(t)$ available to leechers in ISP i belonging to swarm s. We denote the cache bandwidth allocation of ISP i at time t by the vector $\kappa_i(t) = (\kappa_{i,1}(t), \ldots, \kappa_{i,S}(t))$, and the set of feasible cache bandwidth allocations of ISP i by $\mathcal{K}_i = \{\kappa_i | \sum_{s\in\mathcal{S}} \kappa_{i,s} \leq K_i\} \subseteq [0, K_i]^{|\mathcal{S}|}$. We also make the reasonable assumption that $\kappa_{i,s}(t) > 0$ for a swarm s only if the corresponding file is at least partially cached at ISP i at time t.

Given the set $\mathcal{K}_i$ of feasible cache bandwidth allocations for ISP i, a cache bandwidth allocation *policy* π defines $\kappa_i(t)$ as a function of the system's history up to time t, i.e., $(Z(u))_{u<t}$, and past cache allocations $(\kappa_i(u))_{u<t}$. We denote the set of all cache bandwidth allocation policies by Π.

3.2 Caching and System Dynamics

Consider a policy π implemented by ISP i. We model the evolution of the swarms' state by an $I \times S \times 2$ dimensional continuous-time Markov jump process $\mathcal{Z}^\pi = \{Z(t), t \geq 0\}$, which is a collection of S coupled $I \times 2$ dimensional continuous-time Markov jump processes $\mathcal{Z}^\pi_s = \{Z_s(t), t \geq 0\}$.

Consider now a swarm $s \in \mathcal{S}$ under policy π, and denote the transition intensity from state z_s to state z'_s by $q^\pi_{z_s,z'_s}$. Denote by e_i the I dimensional vector whose i^{th} component is 1. The transition intensities from state $z_s = (x_s, y_s)$ are $q^\pi_{z_s,(x_s+e_i,y_s)} = \lambda_{i,s}$ (leecher arrival), $q^\pi_{z_s,(x_s-e_i,y_s)} = \theta x_{i,s}$ (leecher abort), and $q^\pi_{z_s,(x_s,y_s-e_i)} = \gamma y_{i,s}$ (seed departure). The transition

Parameter	Definition
$\mathcal{I}, \mathcal{S}$	Set of ISPs and set of swarms, respectively
$\kappa_{i,s}$	Cache bandwidth allocation of ISP i to swarm s
$\lambda_{i,s}$	Arrival rate of leechers to swarm s in ISP i
θ	Abort rate of leechers
γ	Departure rate of seeds
η	Effectiveness of file sharing
μ, c	Peer upload and download capacity, respectively
$X_{i,s}(t)$	Number of leechers in ISP i in swarm s at time t
$Y_{i,s}(t)$	Number of seeds in ISP i in swarm s at time t
$u_{i,s}^{PL}(t)$	Upload rate in ISP i available to all leechers in swarm s

Table 1: Frequently used notation

intensity to state $(x_s - e_i, y_s + e_i)$, called the download completion rate, is a function of the maximum download rate of the leechers, and the available upload rate to leechers in ISP i.

3.2.1 The case of no cache

Without a cache ($K_i = 0$) the leechers in ISP i would get a share $x_{i,s}/\sum_i x_{i,s}$ of the total upload rate $u_s^P = \sum_i u_{i,s}^P$ [25, 26, 28, 29]. The download completion rate in this case can be expressed as

$$q^{\pi}_{(x_s,y_s),(x_s-e_i,y_s+e_i)} = \min(cx_{i,s}, u_s^P x_{i,s}/\textstyle\sum_i x_{i,s}). \tag{1}$$

We refer to the process defined this way as the *uncontrolled* stochastic process, and we denote it by $\mathcal{Z}$.

3.2.2 The case of cache

Consider that the instantaneous cache bandwidth allocated to swarm s is $\kappa_{i,s}$. The cache bandwidth increases the available upload rate, so that the download completion rate becomes

$$q^{\pi}_{(x_s,y_s),(x_s-e_i,y_s+e_i)} = \min(cx_{i,s}, u_s^P x_{i,s}/\textstyle\sum_i x_{i,s} + \kappa_{i,s}). \tag{2}$$

Since the cache bandwidth allocation can influence the transition intensities of the stochastic process, we refer to $\mathcal{Z}^{\pi}$ as the *controlled* stochastic process. Table 1 summarizes the notation used in the paper.

4 The Optimal Cache Bandwidth Allocation Problem and Stationary Policy

In this section we formulate the optimal cache bandwidth allocation problem and we show the existence of an optimal stationary policy.

The primary goal of ISP i when allocating cache bandwidth to swarm s is to decrease the inter-ISP traffic. Cache bandwidth allocation inherently affects the upload rate available to the leechers, and hence, it can affect the evolution of the process $\mathcal{Z}_s^\pi$.

Let us denote by $I_{i,s}(Z_s(t), \kappa_{i,s}(t))$ the rate of the incoming inter-ISP traffic in ISP i due to swarm s as a function of the cache bandwidth $\kappa_{i,s}(t)$ allocated to swarm s by ISP i and the swarm's state $Z_s(t)$. $I_{i,s}(Z_s(t), \kappa_{i,s}(t))$ also depends on $\kappa_{j,s}(t)$ of ISPs $j \neq i$, but as we focus on the bandwidth allocation problem of ISP i, for simplicity we assume that $\kappa_{j,s}(t) = \kappa_{j,s}$ constant.

We can express the expected amount of incoming inter-ISP traffic under policy $\pi \in \Pi$ from time $t = 0$ until time T as

$$C_i^\pi(z,T) = E_z^\pi \left[\int_0^T \sum_{s \in \mathcal{S}} I_{i,s}(Z_s(t), \kappa_{i,s}(t)) dt \right],$$

where E_z^π denotes the expectation under policy π with initial state $Z(0) = z$.

Given the set Π of feasible cache bandwidth allocation policies, we define the cache bandwidth allocation problem for ISP i as finding the cache bandwidth allocation policy $\pi^* \in \Pi$ that minimizes the average incoming inter-ISP traffic rate $C_i^\pi(z)$ due to P2P content distribution, that is

$$\inf_{\pi \in \Pi} C_i^\pi(z) = \inf_\pi \limsup_{T \to \infty} \frac{1}{T} C_i^\pi(z,T). \tag{3}$$

Consequently, the optimal cache bandwidth allocation problem can be modeled as a continuous-time Markov decision process (MDP) with the optimality criterion defined in (3).

4.1 Optimal Cache Bandwidth Allocation

The first two fundamental questions that we are to answer are (i) whether there is an optimal cache bandwidth allocation policy π^* that solves (3), and (ii) whether there is an optimal policy whose choices only depend on the *current* system state $Z(t)$. Such a policy is called *stationary*. In general, an optimal stationary policy might not exist for a MDP when the action space or the state space is infinite. The following theorem shows that for the cache bandwidth allocation problem there exists an optimal stationary policy.

Theorem 1. *There exists an optimal stationary policy π^* that minimizes the average traffic $C_i^\pi(z)$ of ISP i.*

Proof. Recall that the controlled processes $\mathcal{Z}_s^\pi$ are coupled through the bandwidth allocation policy π. In the following we define four criteria *C1-C4* for $\mathcal{Z}^\pi$ and we use them to prove the theorem.

C1: The set $\mathcal{K}_i$ of cache bandwidth allocations is compact.

C2: For every state $z = (x,y)$ the incoming inter-ISP traffic rate $\sum_s I_{i,s}(z_s, \kappa_{i,s})$ and the transition intensities $(q^{\pi}_{(x_s,y_s),(x_s-e_i,y_s+e_i)})_{s\in\mathcal{S}}$ are continuous functions of $\kappa_{i,s}$.

C3: Define $H(z) = C_i^{\pi}(z) - C_i^{\pi}(a)$, where a is an arbitrarily chosen state. Then $\sum_{z'} H(z')q^{\pi}_{z,z'}$ is continuous in $\kappa_{i,s}$ for every state z.

C4: The average inter-ISP traffic $C_i^{\pi}(z)$ is finite for every policy π and initial state z.

We now formulate the following Lemma based on (Theorem 5.9 in [35]).

Lemma 1. *For a continuous-time MDP with countably infinite state space and non-negative cost, under* C1-C4 *there exists a stationary policy* π^* *that is average cost optimal.*

Since the cost function $C_i^{\pi}(z)$ defined in (3) is the average cost, in order to prove the theorem it is sufficient to show that $\mathcal{Z}^{\pi}$ fullfills the criteria *C1-C4*.

Proof of C1-C3: *C1* follows from $0 \leq \kappa_{i,s}(t) \leq K_i < \infty$. $\sum_s I_{i,s}(z_s, \kappa_{i,s})$ is continuous by assumption, the continuity of the transition intensities $(q^{\pi}_{(x_s,y_s),(x_s-e_i,y_s+e_i)})_{s\in\mathcal{S}}$ w.r.t $\kappa_{i,s}$ follows from (2). *C3* follows from the finiteness of $C_i^{\pi}(z)$ and from *C2*.

Proof of C4: In order to show the finiteness of the average inter-ISP traffic $C_i^{\pi}(z)$ for every policy π and initial state z, we show that $\mathcal{Z}_s^{\pi}$ satisfies the Foster-Lyapunov condition for every $s \in \mathcal{S}$, then we give a bound on the inter-ISP traffic rate in every state of the system. Let us define the Lyapunov function $w(z_s) = \sum_i (x_{i,s} + y_{i,s}) + 1$. Also, let us define the sequence $(t_n)_{n\geq 0}$ of time instants, which consists of the transition epochs of the process and of the instants when $\kappa_{i,s}(t)$ changes according to the policy π. Finally, we define the generalized average drift

$$AW(z_s) = E[w(Z_s(t_{n+1})) - w(Z_s(t_n))|Z_s(t_n) = z_s]. \tag{4}$$

Consider now the Foster-Lyapunov average drift condition [36]

$$|AW(z_s)| < \infty \quad \forall z_s, \text{ and } \; AW(z_s) < -\varepsilon \quad z_s \notin C, \tag{5}$$

where $\varepsilon > 0$ and $C \subset \mathbb{N}_0^{|\mathcal{I}|\times 2}$ is finite. For $\lambda_s < \infty$ the uncontrolled process $\mathcal{Z}_s$ satisfies (5): $|AW(z_s)| \leq 1$ due to the random-walk structure of the process, and $AW(z_s) = (\lambda_s - \theta x_s - \gamma y_s)/(-q_{z_s,z_s}) < -\varepsilon$ for x_s or y_s sufficiently big. Consider now the mean drift $AW^{\pi}(z_s)$ of the controlled process. Again, $|AW^{\pi}(z_s)| \leq 1$. Furthermore we have

$$AW^{\pi}(z_s) \leq AW(z_s)\frac{-q_{z_s,z_s}}{-q_{z_s,z_s} - K_i} < -\varepsilon\frac{-q_{z_s,z_s}}{-q_{z_s,z_s} - K_i} < 0.$$

Consequently, the controlled process $\mathcal{Z}_s^{\pi}$ also satisfies the Foster-Lyapunov average drift condition. Since the process is aperiodic and irreducible, the drift condition guarantees ergodicity [36]. Furthermore, for $\tilde{M} = c > 0$ it holds that $I_{i,s}(z_s, \kappa_{i,s}) \leq \tilde{M}w(z_s)$. This together with the ergodicity of all $\mathcal{Z}_s^{\pi}$ implies that $C_i^{\pi}(z)$ is finite and concludes the proof. $\square$

A consequence of Theorem 1 is that the optimal bandwidth allocation policy π^* is such that the allocation $\kappa_i(t)$ is only a function of the system state $Z(t)$, hence it is constant between the state transitions of $\mathcal{Z}^{\pi*}$.

The optimal policy π^* can be found using the policy iteration algorithm [35], but it requires the solution of the steady state probabilities of the controlled Markov processes $\mathcal{Z}^{\pi}$. This can be prohibitive even for a moderate number of ISPs and swarms. In the next section we propose and discuss different approximations.

5 Cache Bandwidth Allocation Policies

In this section we first discuss a baseline for bandwidth sharing. We then describe three approximations to the optimal cache bandwidth allocation policy.

Throughout the section we assume that the inter-ISP traffic functions $I_{i,s}(z_s, \kappa_{i,s})$ are known, and are continuous convex non-increasing functions of $\kappa_{i,s}$. The assumptions of continuity, convexity and non-increasingness are rather natural.

5.1 Demand-driven Bandwidth Sharing (DDS)

As a baseline for comparison, consider that ISP i does *not* actively allocate its cache bandwidth K_i, therefore leechers at different swarms compete with one another for cache bandwidth. The cache in ISP i maintains a drop-tail queue to store the requests received from the leechers in ISP i, and serves the requests according to a first-in-first-out (FIFO) policy at the available upload bandwidth K_i. Let us denote by $\alpha_{i,s}$ the rate at which leechers of swarm s in ISP i request data from the cache in ISP i, and denote by $\sigma_{i,s}$ the mean service time of these requests. Then the offered load of swarm s to the cache is $\rho_{i,s} = \alpha_{i,s}\sigma_{i,s}$. Clearly, if $\rho_{i,s} \geq 1$ then the FIFO queue is in a blocking state with probability $p_i^b > 0$.

If the requests from leechers in every swarm arrive according to a Poisson process, then the aggregate arrival process is Poisson. Since the arrival process is Poisson, an arbitrary request is blocked (i.e., dropped) with probability $p_{i,s}^b = p_i^b$ despite the possibly heterogeneous mean service times due to the PASTA property [37]. The effective (i.e., not blocked) load for swarm s can be expressed as $(1-p_i^b)\rho_{i,s}$, and consequently the share of cache bandwidth used to serve requests for swarm s can be estimated as

$$\frac{\kappa_{i,s}}{\sum_{s\in\mathcal{S}} \kappa_{i,s}} = \frac{(1-p_i^b)\rho_s}{\sum_{s\in\mathcal{S}} (1-p_i^b)\rho_s} = \frac{\rho_s}{\sum_{s\in\mathcal{S}} \rho_s}. \tag{6}$$

In general, if the arrival process of requests is not Poisson then (6) does not hold. Nevertheless, as under the assumption of a Poisson request arrival process the cache bandwidth is shared among the swarms proportional to the offered load (demand) of the swarms, we refer to this policy as the *demand-driven sharing* (*DDS*) policy.

5.2 One-step Look Ahead Allocation Policy (OLA)

The one-step look ahead (OLA) policy π^{OLA} is a simple approximation of the optimal stationary cache bandwidth allocation policy π^*.

Consider the controlled Markov process $\mathcal{Z}^{\pi^{OLA}}$, and let us denote the n^{th} transition epoch of the process by t_n. Then according to the OLA policy the cache bandwidth allocation $\kappa_i(t)$ of ISP i for $t_n < t \leq t_{n+1}$ is such that it minimizes the incoming inter-ISP traffic rate given the state $Z(t_n) = z$ of the process $\mathcal{Z}^{\pi^{OLA}}$

$$\kappa_i(t) = \arg\min_{\kappa_i \in \mathcal{K}_i} \sum_{s \in \mathcal{S}} I_{i,s}(z_s, \kappa_{i,s}). \tag{7}$$

By following the OLA policy the ISP minimizes the incoming inter-ISP traffic in every state of the process $\mathcal{Z}^{\pi^{OLA}}$. The OLA policy *adapts* to the system state, but unlike the optimal policy π^*, it does not consider the impact of cache bandwidth allocation on the evolution of the number of peers.

Recall that, by assumption, $I_{i,s}(z_s, \kappa_{i,s})$ are continuous convex non-increasing functions of $\kappa_{i,s}$ for every state z_s. In order to obtain the optimal solution to (7) consider the Lagrangian

$$L(z, \kappa_i, \zeta) = \sum_{s \in \mathcal{S}} I_{i,s}(z_s, \kappa_{i,s}) - \zeta\left(\sum_{s \in \mathcal{S}} \kappa_{i,s} - K_i\right), \tag{8}$$

where $\zeta \leq 0$ is the Lagrange multiplier. Then

$$\frac{\partial L(z, \kappa_i, \zeta)}{\partial \kappa_{i,s}} = \frac{\partial I_{i,s}(z_s, \kappa_{i,s})}{\partial \kappa_{i,s}} - \zeta \quad \text{and} \quad \frac{\partial L(z, \kappa_i, \zeta)}{\partial \zeta} = K_i - \sum_{s \in \mathcal{S}} \kappa_{i,s}. \tag{9}$$

Hence, a minimum of L over $\mathcal{K}_i$ is characterized by

$$\begin{aligned} \kappa_{i,s} > 0 \quad &\Rightarrow \quad \frac{\partial_+ I_{i,s}(z_s, \kappa_{i,s})}{\partial \kappa_{i,s}} \geq \zeta \geq \frac{\partial_- I_{i,s}(z_s, \kappa_{i,s})}{\partial \kappa_{i,s}} \\ \kappa_{i,s} = 0 \quad &\Rightarrow \quad \frac{\partial_- I_{i,s}(z_s, \kappa_{i,s})}{\partial \kappa_{i,s}} \geq \zeta, \end{aligned}$$

where ∂_+ and ∂_- denote the right and the left derivative of a semi-differentiable function. Since $\mathcal{K}_i$ is compact and convex, such a minimum exists and can be found using a projected subgradient method [38].

An important insight from the OLA policy is the following. If $I_{i,s}(z_s, \kappa_{i,s})$ are continuously differentiable then at optimality every swarm with non-zero cache bandwidth allocation provides equal marginal traffic saving. If $I_{i,s}(z_s, \kappa_{i,s})$ are not continuously differentiable, then for swarms with non-zero cache bandwidth allocation the intersection of the subdifferentials is non-empty.

5.3 Steady-state Optimal Allocation Policy (SSO)

The opposite of the *OLA* policy is to focus on the long-term evolution of the controlled Markov process $\mathcal{Z}^{\pi}$, that is, on the incoming inter-ISP traffic in steady-state and to consider time-independent cache bandwidth allocation policies $\overline{\pi} = \kappa_i$.

Let us denote the expected number of leechers and seeds in steady-state as a function of the cache bandwidth allocation policy $\overline{\pi}$ by $\overline{x}_{i,s}^{\pi}$ and by $\overline{y}_{i,s}^{\pi}$, respectively. They were shown to be a function of the cache upload rate $\kappa_{i,s}$ allocated to swarm s [13, 14]. As long as the total available upload rate is less than or equal to the total download rate of the leechers

$$\overline{x}_{i,s}^{\pi} = \frac{\lambda_{i,s}}{\nu\left(1+\frac{\theta}{\nu}\right)} - \frac{\kappa_{i,s}}{\mu\eta\left(1+\frac{\theta}{\nu}\right)} - \Delta_i(\mathbf{x},\mathbf{y},\kappa) \tag{10}$$

$$\overline{y}_{i,s}^{\pi} = \frac{\lambda_{i,s}}{\gamma\left(1+\frac{\theta}{\nu}\right)} + \frac{\kappa_{i,s}\theta}{\mu\eta\gamma\left(1+\frac{\theta}{\nu}\right)} + \frac{\theta}{\gamma}\Delta_i(\mathbf{x},\mathbf{y},\kappa), \tag{11}$$

where $\frac{1}{\nu} = \frac{1}{\eta}\left(\frac{1}{\mu} - \frac{1}{\gamma}\right) \geq 0$ [13,26] and

$$\Delta_i(\mathbf{x},\mathbf{y},\kappa) = \frac{\sum_{j\in\mathcal{I}}\left(\lambda_{i,s}\kappa_{j,s} - \kappa_{i,s}\lambda_{j,s}\right)}{\eta\gamma\left(1+\frac{\theta}{\nu}\right)\left(\sum_{j\in\mathcal{I}}\left(\lambda_{j,s} - \kappa_{j,s}\right)\right)}. \tag{12}$$

Otherwise, when the total upload rate exceeds the total download rate, increasing the cache bandwidth allocated to the swarm does not affect the number of leechers and seeds in steady-state, which now depends on the peers' download capacity c [13,26]

$$\overline{x}_{i,s}^{\pi} = \frac{\lambda_{i,s}}{c\left(1+\frac{\theta}{c}\right)} \qquad \overline{y}_{i,s}^{\pi} = \frac{\lambda_{i,s}}{\gamma\left(1+\frac{\theta}{c}\right)}. \tag{13}$$

It is easy to verify that $\frac{\partial \overline{x}_{i,s}}{\partial \kappa_{i,s}} \leq 0$ and that $\frac{\partial^2 \overline{x}_{i,s}}{\partial {\kappa_{i,s}}^2} \geq 0$ for $\kappa_{i,s} \geq 0$, that is, the number of leechers in swarm s in ISP i in steady-state is a convex non-increasing function of the cache bandwidth allocated to swarm s in ISP i.

Given the functions $\overline{x}_{i,s}^{\pi}$ and $\overline{y}_{i,s}^{\pi}$ the steady-state optimal (*SSO*) bandwidth allocation policy can be formulated as

$$\overline{\pi}^* = \underset{\kappa_i\in\mathcal{K}_i}{\arg\min} \sum_{s\in\mathcal{S}} \overline{I}_{i,s}(\kappa_{i,s}), \tag{14}$$

where $\overline{I}_{i,s}(\kappa_{i,s})$ is the incoming inter-ISP traffic rate for the number of leechers and seeds in steady-state.

Since by assumption $I_{i,s}(z_s,\kappa_{i,s})$ is convex non-increasing in $\kappa_{i,s}$ for every state z_s, the steady-state optimal policy $\overline{\pi}^*$ can be found in a similar way as the OLA policy. The difference is that $\overline{I}_{i,s}(\kappa_{i,s})$ is a function of $\kappa_{i,s}$, $\overline{x}_{i,s}^{\pi}$ and $\overline{y}_{i,s}^{\pi}$, and the latter are themselves functions of $\kappa_{i,s}$. Note that the steady-state optimal policy $\overline{\pi}^*$ is not equivalent to the optimal policy π^* of the MDP, as the cache bandwidth allocated to a swarm s in ISP i would be nonzero even when $x_{i,s}(t) = 0$, which happens with nonzero probability.

5.4 Smallest-ratio Priority Allocation

The *SSO* policy exclusively focuses on the long term evolution of the process $\mathcal{Z}^{\pi}$ by minimizing the incoming inter-ISP traffic rate at steady state. It is time independent, i.e. it does not adapt to the current state of the system. The *OLA* policy instead, adapts to the system state by minimizing the instantaneous incoming inter-ISP traffic rate. Nevertheless, the *OLA* policy disregards how cache bandwidth allocation affects the long term evolution of the system.

In the following we use the incoming inter-ISP traffic model in [14] to derive an adaptive cache bandwidth allocation policy that approximates the SSO policy.

5.4.1 Incoming inter-ISP Traffic Model

We first reproduce the incoming inter-ISP traffic model for completeness. As the model is for a single swarm, we omit the subscript s for clarity. The model is based on two assumptions. First, leechers compete with each other for the available upload rate as long as they would be able to download at a higher rate. Second, given a single byte downloaded in ISP i, the distribution of its sources is proportional to the amount of upload rate exposed to the leechers that are located in ISP i.

The leechers in ISP i demand data at a total rate of cx_i. As the cache appears as an arbitrary peer to the leechers in ISP i, the demand is directed to the upload rate κ_i of ISP i's cache and to the publicly available upload rate $u_i^{PL} + \sum_{j \neq i} u_j^P$ of all ISPs. The leechers demand from the cache's upload rate with a probability proportional to its value, i.e, with probability $\kappa_i / (u_i^{PL} + \sum_{j \neq i} u_j^P + \kappa_i)$. The rest they demand from the publicly available upload rate, so the rate D_i^d that leechers in ISP i demand from the publicly available upload rate can be expressed as

$$D_i^d = cx_i \left(1 - \frac{\kappa_i}{u_i^{PL} + \sum_{j \neq i} u_j^P + \kappa_i} \right). \tag{15}$$

If the system is limited by the download rate of the leechers, then the leechers receive the demanded rate. If the system is limited by the available upload rate, then the rate at which the leechers receive is proportional to the total publicly available upload rate divided by the total demanded rate

$$D_i^r = D_i^d min \left(1, \frac{\sum_j u_j^P}{\sum_j D_j^d} \right). \tag{16}$$

The rate that the leechers receive can originate from any ISP. Using the assumption that for a single byte downloaded in ISP i, the distribution of its sources is proportional to the amount of upload rate exposed to leechers in ISP i we get the following estimate of the incoming inter-ISP traffic of ISP i

$$I_i(z_s, \kappa_i) = D_i^r \left(\frac{\sum_{j \neq i} u_j^P}{u_i^{PL} + \sum_{j \neq i} u_j^P} \right). \tag{17}$$

$I_i(z_s, \kappa_i)$ defined by (15) to (17) is a continuous convex non-increasing function of the cache bandwidth κ_i allocated by ISP i.

5.4.2 Smallest-ratio Priority Allocation

Our approximation of the *SSO* policy is based on the results in [13, 14], which show that the dynamics of swarm s in ISP i only depend on the aggregate arrival intensity of leechers $\sum_{j\neq i}\lambda_{j,s}$ and on the aggregate cache capacity $\sum_{j\neq i}\kappa_{j,s}$ in the rest of the ISPs. This observation allows us to focus on a single swarm spread over two ISPs, $\mathcal{I}=\{1,2\}$. ISP 1 is the tagged ISP and ISP 2 is the aggregation of all other ISPs in the network. We denote the ratio of the arrival rates in the two ISPs by $r=\lambda_2/\lambda_1$.

Our focus will be on how the partial derivative $\frac{\partial \bar{I}_1(\kappa_1)}{\partial \kappa_1}$ of the steady-state inter-ISP traffic depends on r. For small κ_1 the incoming inter-ISP traffic $I_1(z,\kappa_1)$ of ISP 1 defined by (15) to (17) can be approximated by

$$I_1(z,\kappa_1)\approx\frac{x_1}{x_1+x_2}u_2^P. \tag{18}$$

We consider the case when the system is limited by the available upload rate, so we substitute (10) and (11) into (18) to obtain an approximation of the steady-state incoming inter-ISP traffic $\bar{I}_1(\kappa_1)$ of ISP 1 as a function of the cache bandwidth. Consider now the derivative at $\kappa_1=0$ and $\kappa_2=0$

$$\frac{\partial \bar{I}_1(\kappa_1)}{\partial \kappa_1}\Big|_{\substack{\kappa_1=0\\ \kappa_2=0}}=-\frac{r^2(\gamma+\nu)(\gamma-\mu)-r\mu(\theta-\gamma)}{(1+\frac{\theta}{\nu})\mu\eta\gamma^2(1+r)^2}. \tag{19}$$

Recall that $\gamma-\mu>0$ is a necessary condition for the upload rate to be the limit, and it implies $\nu>0$ [13, 26]. Hence for $\theta-\gamma\leq 0$, (19) is negative and decreases monotonically in r.

For $\theta-\gamma>0$ we have to consider the mixed second order partial derivative at $\kappa_1=0$ and $\kappa_2=0$

$$\frac{\partial^2 \bar{I}_1(\kappa_1)}{\partial \kappa_1 \partial r}\Big|_{\substack{\kappa_1=0\\ \kappa_2=0}}=-\frac{2r(\gamma+\nu)(\gamma-\mu)+(r-1)\mu(\theta-\gamma)}{(1+\frac{\theta}{\nu})\mu\eta\gamma^2(1+r)^3}. \tag{20}$$

Since $\theta-\gamma>0$, (20) is negative for $r\geq 1$. Consequently allocating cache bandwidth to swarms with a higher ratio r of arrival rates leads to a faster decrease of the steady-state inter-ISP traffic. At the same time, due to the term $(1+r)^3$ in the denominator $\lim_{r\to\infty}\frac{\partial^2 \bar{I}_1(\kappa_1)}{\partial \kappa_1 \partial r}|_{\kappa_1=0}=0$, i.e, swarms with a high arrival ratio r provide approximately the same gain.

This approximation suggests that a priority-based policy that assigns the highest priority to the swarms with highest ratio $r=\lambda_2/\lambda_1$ would resemble the *SSO* allocation policy for small cache bandwidths. We use this insight to define the *smallest-ratio priority* (*SRP*) cache bandwidth allocation policy. Under *SRP* the priority of a swarm is calculated based on the instantaneous ratio of the local leechers to the number of peers in the overlay outside of ISP i, $\hat{r}_{i,s}=\frac{x_{i,s}(t)}{\sum_{j\neq i}z_{j,s}(t)}$. The priority of swarms with $\hat{r}_{i,s}=0$ and $\hat{r}_{i,s}=\infty$ is lowest, and the priorities of the remaining swarms are assigned in decreasing order of the ratios $\hat{r}_{i,s}$. and the priority is inverse proportional to $\hat{r}_{i,s}$ otherwise.

Practical Considerations

ISP i requires global information on the system state in order to compute any of the active cache bandwidth allocation policies presented above. The calculation of the *SSO* allocation relies on the incoming inter-ISP traffic rate for the number of peers in steady-state. In order to compute (10) and (11), ISP i needs to estimate the arrival rates of leechers to the different swarms, i.e. $\lambda_{i,s}$ $\forall s \in \mathcal{S}$, $\forall i \in \mathcal{I}$. The *OLA* policy assumes that the incoming inter-ISP traffic function $I_{i,s}(z_s, \kappa_{i,s})$ is known, furthermore it requires knowledge of the total number of peers in each swarm z_s $\forall s \in \mathcal{S}$. Similarly, under *SRP*, the priority of a swarm is calculated based on the number of local leechers $x_{i,s}$ and on the number of peers z_s in the overlay outside ISP i. In lack of interaction between ISPs and the P2P overlay, ISP i can collect information about the number of local peers in each swarm $z_{i,s}(t)$, and can estimate the aggregate number of peers outside its network $\sum_{j \neq i} z_{j,s}(t)$ by interrogating the tracker. If ISP-overlay interaction is possible [15], e.g., through an ALTO-like service [39], then ISP i could use the service to obtain information about the number of local and remote peers in each swarm. Cache bandwidth optimization could thus be a potential use-case for services like ALTO.

6 Performance Evaluation and Insights

In this section we use simulations and experiments to compare the three approximate cache bandwidth allocation policies to DDS, and to provide insight into the characteristics of an optimal cache bandwidth allocation policy.

6.1 Performance Evaluation Setup

In the following we describe the simulation and experimental settings and the implementation of the different cache bandwidth allocation policies. In both simulations and experiments we consider $I = 2$ ISPs, and use a BitTorrent seed with upload bandwidth K_1 as cache of ISP 1. The cache joins all swarms, but uploads only to leechers in ISP 1. The different cache bandwidth allocation policies are implemented in the peer.

6.1.1 Simulation setup

We used the P2P simulation and prototyping tool-kit ProtoPeer and the corresponding library for BitTorrent [40, 41] for the simulations. The simulations are flow-level: data transmissions are flows and the bandwidth for each flow is calculated according to the max-min-fair-share principle [42], an approximation of the bandwidth sharing behavior of TCP.

We simulate 12 to 26 BitTorrent swarms, each sharing a file of 150MB. The number of swarms is large enough to show the impact of the policies. At the same time, it keeps the run-time of the simulations at a reasonable level of a few hours per simulation run. The peers have an access bandwidth of 1Mbit/s upstream and 16Mbit/s downstream. The peers join swarm s in ISP i according to a Poisson process and, after completing the download, they remain in the swarm for an exponentially distributed seeding time with average $1/\gamma = 10$ minutes.

In order to implement the *OLA* and the *SSO* policies, we use the traffic model in Section 5.4. We implement the *SRP* in the simulator by assigning a priority level to every data flow and by modifying the bandwidth sharing algorithm. The bandwidth of flows with the same priority is calculated according to the original max-min-fair-share algorithm, while flows with a lower priority can only use the link bandwidth not used by flows with higher priority.

6.1.2 Experimental setup

We perform experiments involving approximately 500 Planet-lab nodes using BitTorrent 4.4.0. We scale down the file size, the upload rates and the download rates by a factor of 43 compared to the simulations in order to avoid interfering with other Planet-lab traffic: the file size is 3.5MB, and the upload and download bandwidths of the peers are 23kbit/s and 373kbit/s, respectively.

For every swarm we assign every Planet-lab node to one of the two ISPs, and measure the traffic exchanged between peers belonging to different ISPs. We use one peer per swarm as the cache of ISP 1; these 12 peers run on a dedicated Linux computer. We implement the cache bandwidth allocation policies using hierarchical token bucket (HTB) queues in Linux traffic control. We use one filter per swarm to redirect the upload traffic of the 12 peers to a HTB class that enforces the total cache upload bandwidth limit K_1. For the *SSO* and the *SRP* policies we attach to this class one subclass per swarm. By default each subclass has 500B/s of guaranteed bandwidth in order to keep the TCP connections alive. The actual priority and guaranteed bitrate are then set according to the cache bandwidth allocation policy. The excess bandwidth is distributed among the swarms as defined by the HTB queue. For *SRP* we update the priorities every 10 seconds based on the average number of leechers and seeds over the preceding 30 seconds.

6.2 Stationary Arrival Process

We start by considering the case when peers join swarm s in ISP i according to a stationary Poisson process at a rate of $\lambda_{i,s}$. This corresponds to a system in steady state. Every simulation run corresponds to 6.5 hours of simulated time, and we use the results following a warm-up period of 1.5 hours. For every configuration we show the average of 5 simulation runs together with the 95%-confidence intervals. Every experiment runs for 4 hours, and we use the results after an initial warm-up period of 1 hour.

6.2.1 Cache Bandwidth Allocation Matters

We simulate four scenarios to investigate under what conditions active cache bandwidth allocation can be beneficial. For simplicity, we denote the total arrival rate by $\lambda = \sum_i \sum_s \lambda_{i,s}$.

Scenario	Number of swarms (S)	Identical swarms (s)	$\frac{\lambda_s}{\lambda}$	$\frac{\lambda_{2,s}}{\lambda_{1,s}}$
unif.,1:10	12	1,..,12	1/12	10
zipf,1:10	12		$\propto \frac{1}{s}$	10
unif.,1:1+1:10	12	1,..,10	1/12	10
		11,12	1/12	1
het.,2:2+1:10	15	1,..,4	1/8	10
		5,..,15	1/22	1

Table 2: Relative peer arrival rates in the simulated scenarios.

We use the same total arrival rate $\lambda = 30$/min for all four scenarios, but the four scenarios differ in terms of the arrival rates $\lambda_{i,s}$ of the peers between swarms and between ISPs. Table 2 shows the relative arrival rates for the four scenarios. The ratio $\frac{\lambda_s}{\lambda}$ is related to the size of swarm s compared to all swarms, while $\frac{\lambda_{2,s}}{\lambda_{1,s}}$ is related to the share of local peers in swarm s. In [43], the authors measured the top-AS fraction of different swarms, defined as the maximum number of peers in one AS of the swarm normalized by the size of the swarm ($\max_i \frac{z_i}{\sum_j z_j}$). The top-AS fraction was found to vary from a minimum of slightly less than 0.1, for swarms sharing international content, to a maximum of 0.5, for swarms sharing regional content. We can use these numbers to obtain estimates of the relative arrival intensities of leechers as follows. In absence of a cache ($\kappa_i = 0\ \forall i \in \mathcal{I}$), the numbers of both seeders and leechers are proportional to the arrival rate of leechers to swarms, whether the system is upload rate limited (10-11) or download rate limited (13). Consequently, by substituting $\kappa_i = 0\ \forall i \in \mathcal{I}$ in (10-13), we can use the top-AS fraction to calculate the ratio $\frac{\lambda_{2,s}}{\lambda_{1,s}}$. Using this approximation a top-AS fraction of 0.1 corresponds to $\frac{\lambda_{2,s}}{\lambda_{1,s}}$ slightly greater than 9, and a top-AS fraction of 0.5 corresponds to $\frac{\lambda_{2,s}}{\lambda_{1,s}} = 1$. Given these approximate relative arrival intensities, our evaluation scenarios are constructed so that they allow us to isolate the factors that influence the efficiency of cache bandwidth allocation policies.

As an example, in scenario *unif.,1:1+1:10* all $S = 12$ swarms have the same arrival rate $\lambda_s = \lambda/12$. The arrival rates for swarms 1 to 10 are asymmetric ($\lambda_{2,s} = 10\lambda_{1,s}$), while for swarms 11 and 12 they are symmetric ($\lambda_{2,s} = \lambda_{1,s}$). In scenario *het.,2:2+1:10* the swarms have different arrival rates. 4 out of 15 swarms have an arrival rate of $\lambda_s = \lambda/8$, and are asymmetric, $\lambda_{2,s} = 10\lambda_{1,s}$. The remaining 11 swarms have an arrival rate of $\lambda_s = \lambda/22$ and are symmetric $\lambda_{2,s} = \lambda_{1,s}$. Compared to *unif.,1:1+1:10*, in this scenario the symmetric swarms, though more popular in ISP 1, are less popular in total than the asymmetric ones. The use of Zipf's law for the arrival intensities in scenario *zipf,1:10* is motivated by recent measurements that show that the distribution of the number of concurrent peers over swarms exhibits Zipf like characteristics over a wide range of swarm sizes [44,45]. Symmetric and asymmetric swarms are motivated by measurements that show the difference in

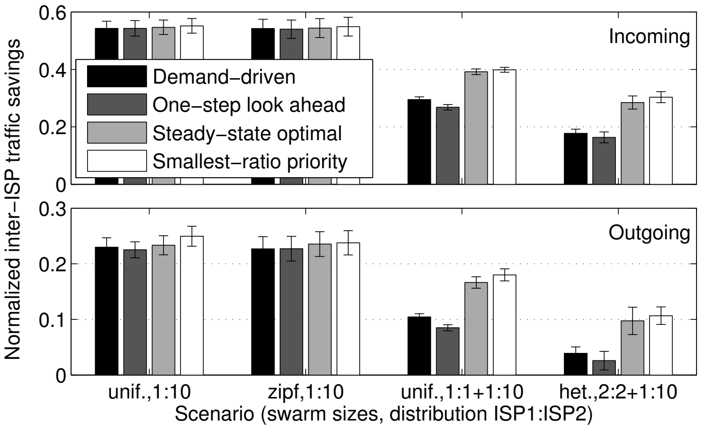


Figure 1: Incoming and outgoing inter-ISP traffic savings for the four scenarios and four policies for $K_1 = 30$Mbit/s. Simulation results.

terms of the spatial distribution of peers between contents of regional and of global interest (e.g., the popularity of movies depending on the language [45]).

Fig. 1 shows the normalized incoming and outgoing inter-ISP traffic saving of ISP 1 for the four scenarios for the *DDS*, *OLA*, *SSO* and *SRP* allocation policies. We calculate the normalized inter-ISP traffic saving as the decrease of the average inter-ISP traffic due to installing a cache divided by the average inter-ISP traffic without a cache ($K_1 = 0$), that is, $(C_i|_{K_1=0} - C_i^{\pi})/C_i|_{K_1=0}$. The upload bandwidth of the cache in ISP 1 is $K_1 = 30$Mbit/s.

For the *unif.,1:10* and the *zipf,1:10* scenarios, in which the ratio $\lambda_{2,s}/\lambda_{1,s} = 10$ is the same for all swarms, the difference between the results for the different cache bandwidth allocation policies is within the confidence interval. However, for the scenarios *unif.,1:1+1:10* and *het.,2:2+1:10* the bandwidth allocation policies make a significant difference in terms of traffic savings, both in terms of incoming and outgoing inter-ISP traffic. These results indicate that cache bandwidth allocation affects the inter-ISP traffic savings when the distribution of the peers over the ISPs is different among swarms, as for the *unif.,1:1+1:10* and the *het.,2:2+1:10* scenarios.

A comparison of the different policies in Fig. 1 for the *unif.,1:1+1:10* scenario reveals that the effect of the *OLA* policy on the inter-ISP traffic saving is opposite to the effect of the *SSO* and the *SRP* policies. The *OLA* policy performs worse than *DDS*, but the *SSO* and *SRP* policies compared to *DDS* drastically increase the incoming and outgoing inter-ISP traffic savings. For the *SSO* policy, the incoming inter-ISP traffic savings increase by about 33 percent and the outgoing inter-ISP traffic savings by over 60 percent. For the *het.,2:2+1:10* scenario the savings increase by 60 and 150 percent, respectively. The *SRP* policy achieves even better gains. For the *het.,2:2+1:10* scenario for example, the savings

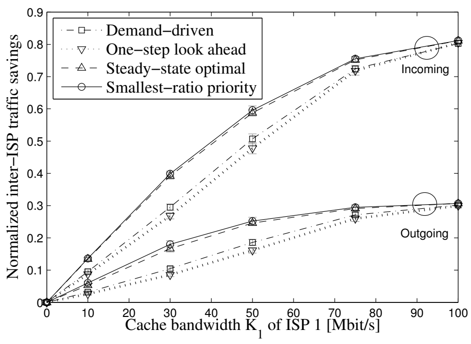


Figure 2: Incoming and outgoing inter-ISP traffic saving for the *unif.,1:1+1:10* scenario vs. cache bandwidth in ISP 1. Simulation results.

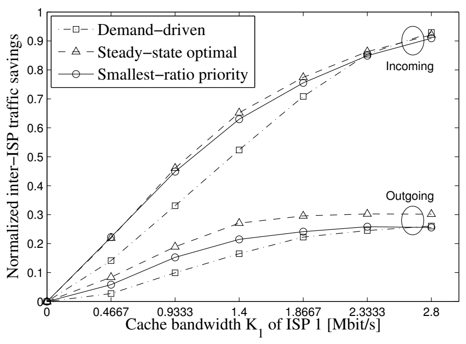


Figure 3: Incoming and outgoing inter-ISP traffic saving for the *unif.,1:1+1:10* scenario vs. cache bandwidth in ISP 1. Experiment results.

increase by 71 percent for the incoming inter-ISP traffic and 172 percent for the outgoing traffic. Considering that P2P cache eviction policies achieve within 10 to 20 percent of the hit rate of the optimal off-line eviction policy [10, 11], the 30 to 70 percent decrease of the incoming inter-ISP traffic achieved through cache bandwidth allocation is more than what could be achieved through improved cache eviction policies.

6.2.2 Inter-ISP traffic savings

Fig. 2 shows the incoming and outgoing inter-ISP traffic savings normalized by the inter-ISP traffic without cache ($K_1 = 0$) for the *unif.,1:1+1:10* scenario, as a function of the cache bandwidth K_1. The figure confirms that the observations made in Fig. 1 hold for a wide range of cache bandwidths K_1. Only above $K_1 \approx 75$Mbit/s, when the available upload bandwidth in ISP 1 exceeds the aggregate download bandwidth of the leechers within ISP 1, the marginal traffic saving diminishes and so does the difference between the policies. We note that the *SRP* policy performs slightly better than the *SSO* policy for all cache bandwidths. This is because the the *SSO* allocation can be far from optimal when the instantaneous number of peers in the system is far away from the steady-state average number of peers. We show the corresponding experimental results in Fig. 3. We omit the results for the *OLA* policy since it performed poorly in all simulated scenario. As shown in Fig. 3, the experimental results match the simulation results (cf. Fig. 2) and confirm the significant gain of cache bandwidth allocation observed in the simulations. The only difference is that the *SRP* policy performs slightly worse than in the simulations, which is due to the impact of the network layer implementation of bandwidth allocation and priorities on TCP congestion control. An application layer implementation of the policy could prevent this.

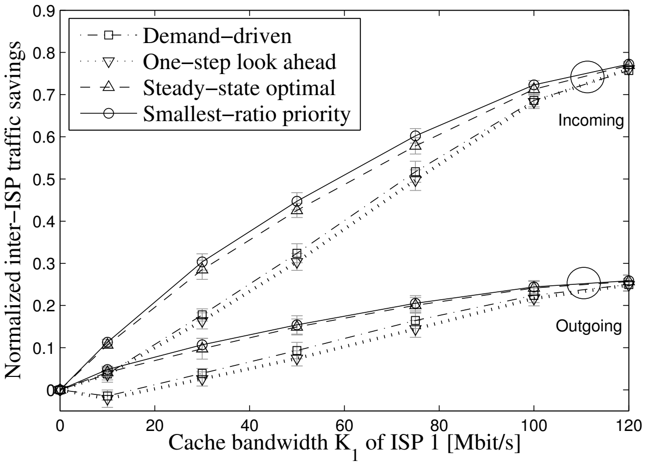


Figure 4: Incoming and outgoing inter-ISP traffic savings for the *het.,2:2+1:10* scenario vs. cache bandwidth in ISP 1. Simulation results.

Fig. 4 shows the incoming and outgoing inter-ISP traffic saving normalized by the inter-ISP traffic without cache ($K_1 = 0$) for the *het.,2:2+1:10* scenario as a function of the cache bandwidth K_1. The figure allows us to draw similar conclusions as Fig. 2, except for the dip in the outgoing inter-ISP traffic saving for *DDS* at $K_1 = 10$Mbit/s. While surprising at first sight, the potential increase of the outgoing inter-ISP traffic due to caching for small, symmetric swarms (i.e., swarms 5 to 15) was pointed out in [13]. Since the *SRP* and the *SSO* policies allow little cache bandwidth to be used by the symmetric swarms for low K_1, they provide outgoing inter-ISP traffic savings even at $K_1 = 10$Mbit/s.

6.2.3 Cache Upload Rate to Swarms

In order to understand how the different policies allocate bandwidth to the different swarms, we show two indifference maps of ISP 1 for the *unif.,1:1+1:10* scenario in Fig. 5 and Fig. 6. The horizontal and the vertical axes show the cache bandwidth allocated to each of the 10 asymmetric ($\lambda_{2,s} = 10\lambda_{1,s}$) and to each of the 2 symmetric ($\lambda_{2,s} = \lambda_{1,s}$) swarms, respectively. The curves show combinations of bandwidth allocations that lead to a particular inter-ISP traffic saving ΔI_1 (i.e., was there no cache bandwidth constraint K_1, ISP 1 would be indifferent between allocations on the same indifference curve). The straight diagonal lines show different cache bandwidth constraints K_1. The *SSO* cache bandwidth allocation for K_1 is given by the coordinates of the point at which the cache bandwidth constraint line for K_1 is tangent to the indifference curve. The dotted line connects all such points: it shows the *SSO* cache bandwidth allocation for different K_1.

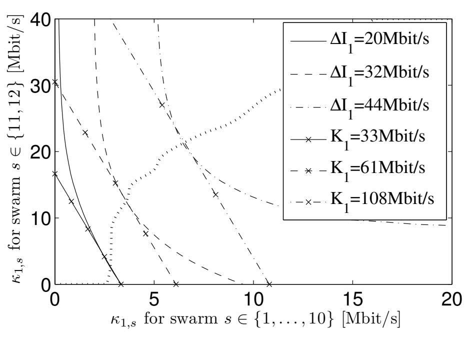


Figure 5: Indifference map of ISP 1 for the *unif.,1:1+1:10* scenario based on simulation results. The dotted line shows the *SSO* cache bandwidth allocation for different values of cache bandwidth K_1.

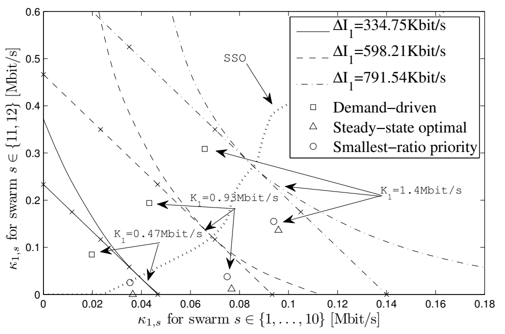


Figure 6: Indifference map of ISP 1 for the *unif.,1:1+1:10* scenario based on the experiments, and the actual average cache upload rates for the *DDS*, *SSO* and *SRP* policies. Experiment results.

Fig. 5 shows the indifference map based on simulation results. We note that for $K_1 \leq$ 30Mbit/s all cache bandwidth should be allocated to the 10 asymmetric swarms, above that, as K_1 increases so does the bandwidth that should be allocated to the 2 symmetric swarms. We also note that the shape of the indifference curves confirms that the inter-ISP traffic saving for a single swarm is a concave non-decreasing function of $\kappa_{i,s}$.
Fig. 6 shows the indifference map and the actual average cache upload rate received by the asymmetric (horizontal) and the symmetric (vertical) swarms under the three allocation policies, based on experiment results. There is one marker per policy and total cache bandwidth K_1. The figure shows how the cache upload rate received by the swarms differs under the three policies depending on the cache bandwidth limit K_1. Under both *SRP* and *SSO* the cache uploads to the symmetric swarms at a significantly lower rate than under *DDS* except for very high K_1. which is the key to the higher inter-ISP traffic savings of both policies.

6.3 Non-Stationary Arrival Process

So far we only looked at a system in steady state. In order to investigate the robustness of the bandwidth allocation policies to the system dynamics, we now turn to the case of a non-stationary arrival process. We consider that the leechers join swarm s in ISP i according to a non-stationary Poisson process with rate

$$\lambda_{i,s}(t) = \lambda_{i,s}^0 e^{-\frac{t}{\tau}}, \tag{21}$$

where $\lambda_{i,s}^0$ is the initial arrival rate and τ is the attenuation parameter of peer arrival rate. (21) has been shown to be a good model of the peer arrival rate during the swarm's lifes-

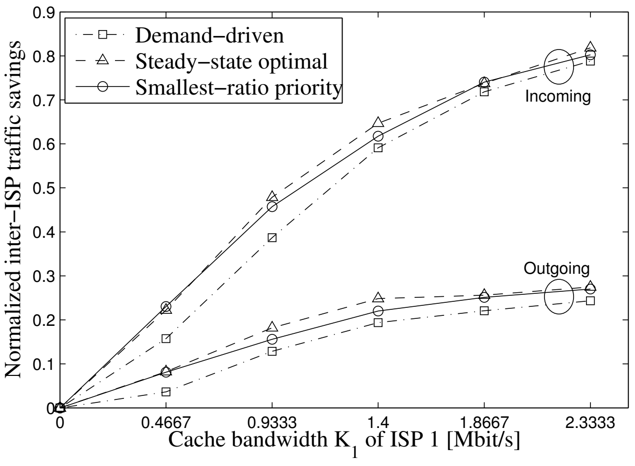


Figure 7: Incoming and outgoing inter-ISP traffic saving for the non-stationary scenario vs. cache bandwidth in ISP 1. Experiment results.

pan in [46, 47]. To derive the inter-arrival times of peers to swarm s, we simulated the non-stationary arrival process using the thinning method by Lewis and Schedler [48]. We considered $\sum_{i\in\mathcal{I}} \lambda_{i,s}^0 = \frac{1}{8}$ and $\tau = 4000$, which result in a swarm lifespan of about 5 hours, and a total peer population during the lifespan of 500 peers. We performed experiments starting a new swarm every 15 minutes, for 6.5 hours. The swarms starting at 0h, 2h, 4h, 6h are symmetric ($\lambda_{2,s} = \lambda_{1,s}$), while the rest of the swarms are asymmetric ($\lambda_{2,s} = 10\lambda_{1,s}$).

Fig. 7 shows the incoming and outgoing inter-ISP traffic saving normalized by the inter-ISP traffic without cache ($K_1 = 0$) for the non-stationary scenario described above, as a function of the cache bandwidth K_1. The inter-ISP traffic savings show a similar trend as under the stationary arrival process (c.f., Fig. 3). Comparing Figures 7 and 3, we observe that the benefit of cache bandwidth allocation is slightly reduced, although still significant: the *SSO* and the *SRP* policies achieve savings in the order of 20 to 30 percent compared to the *DD* allocation. It is important to note that the computation of the *SSO* policy and the derivation of the *SRP* policy in Section 5.4 assume that the system is in steady-state, yet the policies provide significant savings in a non-stationary system.

7 Conclusion

Motivated by the large amount of inter-ISP P2P traffic, we investigated a new dimension of P2P cache resource management, the allocation of cache upload bandwidth between overlays. We formulated the problem of cache bandwidth allocation as a Markov decision process, and showed the existence of an optimal stationary allocation policy. Based on insights obtained from the model, we proposed three bandwidth allocation policies to approximate the optimal allocation policy. We performed simulations and experiments to evaluate the performance of the proposed policies. We demonstrated the importance of capturing the cache's impact on the swarm dynamics for cache bandwidth allocation. We identified the heterogeneity of the swarm's distribution between ISPs as the primary factor that influences the potential traffic savings through cache bandwidth allocation. Our results show that the proposed smallest ratio priority policy can decrease the amount of inter-ISP traffic between 30 to 60 percent, which is significantly higher than what could potentially be achieved exclusively through improved peer-to-peer cache eviction policies.

8 Acknowledgement

The authors thank Alexandre Proutiere for his comments on the drafts of the conference version of this work.

References

[1] H. Schulze and K. Mochalski, "Internet Study 2008/2009," 2009. [Online]. Available: http://www.ipoque.com/resources/internet-studies

[2] P. Eckersley, F. von Lohmann, and S. Schoen, "Packet forgery by ISPs: A report on the Comcast affair," White paper, Nov. 2007.

[3] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P systems co-operate for improved performance?" *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 29–40, Jul. 2007.

[4] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: Provider portal for applications," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 351–362, Oct. 2008.

[5] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in BitTorrent via biased neighbor selection," in *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, Jul. 2006, pp. 66–75.

[6] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 363–374, Oct. 2008.

[7] S. L. Blond, A. Legout, and W. Dabbous, "Pushing bittorrent locality to the limit," *Computer Networks*, vol. 55, no. 3, pp. 541–557, 2011.

[8] OverCache P2P, "http://www.oversi.com."

[9] PeerApp UltraBand, “http://www.peerapp.com.”

[10] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Woźniak, “Cache replacement policies for P2P file sharing protocols,” *Euro. Trans. on Telecomms.*, vol. 15, pp. 559–569, Nov. 2004.

[11] M. Hefeeda and O. Saleh, “Traffic modeling and proportional partial caching for peer-to-peer systems,” *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1447–1460, Dec. 2008.

[12] N. Megiddo and D. Modha, “ARC: A Self-Tuning, Low Overhead Replacement Cache,” in *Proc. of USENIX File & Storage Technologies Conference (FAST)*, 2003, pp. 115 – 130.

[13] F. Lehrieder, G. Dán, T. Hoßfeld, S. Oechsner, and V. Singeorzan, “The impact of caching on BitTorrent-like peer-to-peer systems,” in *Proc. IEEE Int’l Conf. Peer-to-Peer Computing (P2P)*, Aug. 2010.

[14] F. Lehrieder, G. Dan, T. Hossfeld, S. Oechsner, and V. Singeorzan, “Caching for BitTorrent-Like P2P Systems: A Simple Fluid Model and Its Implications,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 1176–1189, 2012.

[15] G. Dán, T. Hoßfeld, S. Oechsner, P. Cholda, R. Stankiewicz, I. Papafili, and G. Stamoulis, “Interaction patterns between P2P content distribution systems and ISPs,” *IEEE Communications Magazine*, vol. 49, no. 5, pp. 222–230, May 2011.

[16] S. Ren, E. Tan, T. Luo, L. Guo, S. Chen, and X. Zhang, “TopBT: A topology-aware and infrastructure-independent BitTorrent,” in *Proc. IEEE INFOCOM*, Apr. 2011.

[17] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit, “Are file swapping networks cacheable? Characterizing P2P traffic,” in *Proc. Int’l Workshop on Web Content Caching and Distribution (WCW)*, Aug. 2002.

[18] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, “Should internet service providers fear peer-assisted content distribution?” in *Proc. of ACM SIGCOMM IMC*, 2005, pp. 63–76.

[19] G. Dan, “Cooperative caching and relaying strategies for peer-to-peer content delivery,” in *Proc. of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2008, pp. 1–16.

[20] J. Dai, B. Li, F. Liu, B. Li, and H. Jin, “On the efficiency of collaborative caching in ISP-aware P2P networks,” in *Proc. IEEE INFOCOM*, Apr. 2011.

[21] I. Papafili, G. D. Stamoulis, F. Lehrieder, B. Kleine, and S. Oechsner, “Cache capacity allocation to overlay swarms,” in *International Workshop on Self-Organizing Systems*, Feb. 2011.

[22] R. S. Peterson and E. G. Sirer, "Antfarm : Efficient Content Distribution with Managed Swarms," in *Proc. of NSDI*, 2009, pp. 107–122.

[23] R. S. Peterson, B. Wong, and E. G. Sirer, "A Content Propagation Metric for Efficient Content Distribution," in *Proc. of ACM SIGCOMM*, vol. 41, no. 4, New York, New York, USA, 2011, pp. 326–337.

[24] A. Sharma, A. Venkataramani, and A. A. Rocha, "Pros & Cons of Model-based Bandwidth Control for Client-assisted Content Delivery," *CoRR*, vol. abs/1209.5, 2012.

[25] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 2242–2252.

[26] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 367–378.

[27] F. Clévenot-Perronnin, P. Nain, and K. W. Ross, "Multiclass P2P networks: Static resource allocation for service differentiation and bandwidth diversity," *Performance Evaluation*, vol. 62, pp. 32–49, Oct. 2005.

[28] Y. Tian, D. Wu, and K. W. Ng, "Modeling, analysis and improvement for BitTorrent-like file sharing networks," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–11.

[29] I. Rimac, A. Elwalid, and S. Borst, "On server dimensioning for hybrid P2P content distribution networks," in *Proc. IEEE Int'l Conf. Peer-to-Peer Computing (P2P)*, Sep. 2008, pp. 321–330.

[30] R. Izhak-Ratzin, H. Park, and M. van der Schaar, "Online Learning in BitTorrent Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2280–2288, 2012.

[31] E. J. Friedman, J. Y. Halpern, and I. Kash, "Efficiency and Nash Equilibria in a Scrip System for P2P Networks," in *Proc. of ACM EC*, 2006, pp. 140–149.

[32] H. Park and M. V. D. Schaar, "A Framework for Foresighted Resource Reciprocation in P2P Networks," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 101–116, 2009.

[33] R. Izhak-Ratzin, H. Park, and M. van der Schaar, "Reinforcement Learning in BitTorrent Systems," in *Proc. of IEEE INFOCOM*, 2011, pp. 406–410.

[34] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurement, analysis, and modeling of BitTorrent-like systems," in *Proc. ACM Internet Measurement Conf. (IMC)*, Oct. 2005, pp. 35–48.

[35] X. Guo and O. Hernandez-Lerma, *Continuous-time Markov Decision Processes*, ser. Springer Series in Stochastic Modelling and Applied Probability. Springer, 2009.

[36] R. L. Tweedie, "Criteria for ergodicity, exponential ergodicity and strong ergodicity of markov processes," *J. Appl. Prob.*, vol. 18, pp. 122–130, 1981.

[37] R. W. Wolff, “Poisson arrivals see time averages,” *Operations Research*, vol. 30, no. 2, pp. 223–231, 1982.

[38] N. Z. Shor, *Minimization Methods for Non-differentiable Functions*, ser. Springer Series in Computational Mathematics. Springer, 1985.

[39] M. Stiemerling, S. Kiesel, S. Previdi, and M. Scharf, “ALTO Deployment Considerations,” Internet Engineering Task Force (IETF), Internet-Draft, 2013.

[40] “Protopeer,” http://protopeer.epfl.ch/index.html.

[41] W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer, “ProtoPeer: A P2P toolkit bridging the gap between simulation and live deployment,” in *Proc. International Conference on Simulation Tools and Techniques*, Mar. 2009.

[42] D. Bertsekas and R. Gallagher, *Data Networks*. Prentice Hall, 1987.

[43] T. Hoß feld, F. Lehrieder, D. Hock, S. Oechsner, Z. Despotovic, W. Kellerer, and M. Michel, “Characterization of BitTorrent Swarms and their Distribution in the Internet,” *Computer Networks*, vol. 55, no. 5, pp. 1197–1215, 2011.

[44] G. Dán and N. Carlsson, “Power-law revisited: A large scale measurement study of P2P content popularity,” in *Proc. Int’l Workshop on Peer-to-Peer Systems (IPTPS)*, April 2010.

[45] T. Hoßfeld, F. Lehrieder, D. Hock, S. Oechsner, Z. Despotovic, W. Kellerer, and M. Michel, “Characterization of BitTorrent swarms and their distribution in the Internet,” *Computer Networks*, vol. 55, no. 5, Apr. 2011.

[46] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, “A Performance Study of BitTorrent-like Peer-toPeer Systems,” *IEEE J. Sel. Areas Commun.*, vol. 25, no. 1, pp. 155–169, 2007.

[47] N. Carlsson, G. Dán, A. Mahanti, and M. Arlitt, “A Longitudinal Characterization of Local and Global BitTorrent Workload Dynamics,” in *Proc. of Passive and Active Measurement Conference (PAM)*, 2012, pp. 252–262.

[48] P. Lewis and G. Shedler, “Simulation of Nonhomogeneous Poisson Processes by Thinning,” *Nav. Res. Logist. Q.*, vol. 26, pp. 403 – 413, 1979.

Paper **C**

Convergence in Player-Specific Graphical Resource Allocation Games

Valentino Pacifici and György Dán

in IEEE Journal on Selected Areas in Communications, December 2012.

Convergence in Player-Specific Graphical Resource Allocation Games

Valentino Pacifici and György Dán [*]

Abstract

As a model of distributed resource allocation in networked systems, we consider resource allocation games played over a influence graph. The influence graph models limited interaction between the players due to, e.g., the network topology: the payoff that an allocated resource yields to a player depends only on the resources allocated by her neighbors on the graph. We prove that pure strategy Nash equilibria (NE) always exist in graphical resource allocation games and we provide a linear time algorithm to compute equilibria. We show that these games do not admit a potential function: if there are closed paths in the influence graph then there can be best reply cycles. Nevertheless, we show that from any initial allocation of a resource allocation game it is possible to reach a NE by playing best replies and we provide a bound on the maximal number of update steps required. Furthermore we give sufficient conditions in terms of the influence graph topology and the utility structure under which best reply cycles do not exist. Finally we propose an efficient distributed algorithm to reach an equilibrium over an arbitrary graph and we illustrate its performance on different random graph topologies.

1 Introduction

Resource allocation is a fundamental problem in communication systems. In general, resource allocation problems are characterized by a set of nodes competing for the same set of resources, and arise in a wide variety of contexts, from congestion control, through content management [1, 2] to dynamic channel allocation for opportunistic spectrum access [3, 4]. The nodes competing for the resources are often autonomous entities, such as end hosts or mobile nodes, and therefore the resource allocation problem has to be solved in a decentralized manner. Furthermore, the solution should be compatible with the nodes' own interests.

[*]The authors are with the Laboratory for Communication Networks, School of Electrical Engineering, and the ACCESS Linnaeus Centre, KTH Royal Institute of Technology. Osquldas väg 10, 10044 Stockholm. Tel.: +46 87904251. E-mail: {pacifici,gyuri}@kth.se. Manuscript received December 15, 2011; revised May 25, 2012; accepted July 15, 2012.

Congestion games are widely used as a game theoretic model of distributed resource allocation [5]. In congestion games a set of players allocate resources in order to maximize their own utility, but the utility provided by an individual resource to a player is a function of how many other players have allocated the resource. In the case of networked systems, the interactions between the players are often limited by the network topology, which can be captured by introducing an influence graph in the model of congestion games [6, 7, 8]. Under certain conditions of symmetry, e.g., homogeneous or linear utility functions [6, 7, 8, 9, 10, 11, 12], congestion games allow a potential function [13], and thus the two most fundamental questions, the existence of pure Nash equilibria and the convergence of myopic learning rules to the equilibria are answered. For congestion games that do not admit a potential function, the answer to these two fundamental questions is, in general, not known [14, 15, 16].

In this work we consider a class of resource allocation games that gives rise to a graphical player-specific congestion game that does not admit a potential function. In the model we consider, the player-specific utility of a resource to a player is amortized if any of its neighbors allocates the same resource. This model captures problems arising in a number of fields: object placement in the context of CPU caching in computer architectures [17], the placement of contents in clean-slate information centric network architectures [1], the problem of cooperative caching between Internet service providers [2], the distributed allocation of radio spectrum to radio transmitters [3, 4], and the distributed scheduling of jobs [18].

Our contributions are the following. We show that Nash equilibria exist in graphical resource allocation games for arbitrary influence graphs and payoff structures, and give a bound on the complexity of finding equilibria. We show that nodes might cycle arbitrarily long before reaching equilibrium if the influence graph is non-complete, thus the game does not admit a potential function. We then provide sufficient conditions in terms of the graph topology and the payoff structure that guarantee that cycles do not exist if players play best replies. Furthermore, we give a sufficient condition under which a simple and efficient distributed algorithm can be used to reach an equilibrium, and illustrate the efficiency of the algorithm with numerical results.

The rest of the paper is structured as follows. We define graphical resource allocation games in Section 2, and provide results for arbitrary graph topologies in Section 3. We then analyze the convergence to equilibria for complete graphs in Section 4 and for specific payoff structures in Section 5. Section 6 introduces the distributed algorithm for reaching equilibrium. In Section 7 we discuss related work, and Section 8 concludes the paper.

2 The Resource Allocation Game

In the following we formulate the problem of graphical resource allocation as a non-cooperative game played on a graph. We consider a set of nodes N and a set of resources $\mathcal{R}$. Every node is located at a vertex of an undirected graph $\mathcal{G} = (N, E)$, called the influence graph. We denote the set of neighbors of node i by $\mathcal{N}(i)$, i.e., $\mathcal{N}(i) = \{j|(i,j) \in E\}$. Each node i allocates $K_i \in \mathbb{N}_+$ different resources: we describe the set of resources allocated by node i with the $|\mathcal{R}|$ dimensional vector $a_i = (a_i^1, \dots, a_i^{|\mathcal{R}|})$, whose component $a_i^r \in \{0,1\}$ is 1 if resource r is allocated by node i. Thus the set of feasible resource allocation vectors for node i is $\mathcal{A}_i = \{a_i | \sum_r a_i^r \leq K_i\} \subseteq \{0,1\}^{|\mathcal{R}|}$. To ease notation we say that a resource $r \in \mathcal{R}$ is *i-busy* if it is allocated by at least one of node i's neighbors and we define $\pi_i^r(a_{-i}^r) \triangleq \prod_{j\in\mathcal{N}(i)} (1 - a_j^r) = 0$, otherwise we say that resource $r \in \mathcal{R}$ is *i-free*.
We call c_{ir} the value of resource r for node i. The payoff $U_i^r(1, a_{-i}^r)$ that a node i gets from allocating a resource r is influenced by the resource allocation of its neighboring nodes $\mathcal{N}(i)$. A node i allocating resource r gets as payoff the value c_{ir} if resource r is *i-free.* If resource r is *i-busy*, node i gets payoff $\delta_i c_{ir}$, where δ_i is the cost of sharing for node i, $0 \leq \delta_i < 1$,

$$U_i^r(1, a_{-i}^r) = \begin{cases} c_{ir} & \text{if } \pi_i^r(a_{-i}^r) = 1 \\ \delta_i c_{ir} & \text{if } \pi_i^r(a_{-i}^r) = 0. \end{cases} \tag{1}$$

A node does not get any payoff from the resources it does not allocate, i.e, $U_i^r(0, a_{-i}^r) = 0$.
We model the resource allocation problem as a strategic game

$$\Gamma =< N, (\mathcal{A}_i)_{i\in N}, (U_i)_{i\in N} >,$$

where the utility function of player i is the sum of the payoffs $U_i(a_i, a_{-i}) = \sum_r U_i^r(a_i^r, a_{-i}^r)$. Note that the influence graph influences the payoffs that player i gets from the allocated resources via the neighbor set $\mathcal{N}(i)$, i.e., the utility of player i is entirely specified by the actions of players $j \in \mathcal{N}(i)$. Under the assumption that every player i allocates always K_i resources, the graphical resource allocation game we consider is a graphical player-specific matroid congestion game.
In the following we outline two applications of graphical resource allocation games.

2.1 Selfish Object Replication on Graphs

Consider a set N of nodes and a set $\mathcal{R}$ of objects of unit size. The demand for object $r \in \mathcal{R}$ at node $i \in N$ is given by the rate $w_i^r \in \mathbb{R}_+$. Node i has integer storage capacity K_i which it uses to replicate objects locally. The marginal cost of serving requests for object r in node i is α_i if the object is replicated in node i, it is β_i if the object is replicated in a node $j \in \mathcal{N}(i)$ neighboring i, and it is γ_i

otherwise. It is reasonable to consider that it is not more costly to access a resource replicated locally than one replicated at a neighbor, and it is less costly to access a resource replicated at a neighbor than retrieving it directly from the common set of resources. Formally $\alpha_i \leq \beta_i < \gamma_i$. The cost of node i due to object r is proportional to the demand w_i^r, and is a function of a_i and the replication states a_{-i} of the neighboring nodes, and its total cost is

$$\begin{aligned} C_i(a_i, a_{-i}) = \quad & \textstyle\sum_{r\in\mathcal{R}} C_i^r(a_i^r, a_{-i}^r) \\ = \quad & \textstyle\sum_{r\in\mathcal{R}} w_i^r \Big[\alpha_i a_i^r + (1-a_i^r)\left[\gamma_i \pi_i^r(a_{-i}^r) + \beta_i(1-\pi_i^r(a_{-i}^r))\right]\Big] \end{aligned} \tag{2}$$

The goal of node i is to choose a replication strategy a_i^* that minimizes its total cost given the strategy profile a_{-i} of the other nodes. Observe that the cost of object r for node i can be expressed as

$$C_i^r(a_i^r, a_{-i}^r) = C_i^r(0, a_{-i}^r) - \big(C_i^r(0, a_{-i}^r) - C_i^r(a_i^r, a_{-i}^r)\big) = C_i^r(0, a_{-i}^r) - CS_i^r(a_i^r, a_{-i}^r),$$

where $CS_i^r(a_i^r, a_{-i}^r)$ is the cost saving that node i achieves through object r given the other nodes' replication strategies. Since the cost $C_i^r(0, a_{-i}^r)$ is independent of the action a_i^r of node i, finding the minimum cost is equivalent to maximizing the aggregated cost saving

$$\begin{aligned} \arg\min_{a_i} C_i(a_i, a_{-i}) \qquad & = \arg\min_{a_i} \sum_r C_i^r(a_i^r, a_{-i}^r) \\ & = \arg\min_{a_i} \Big(\sum_r C_i^r(0, a_{-i}^r) - \sum_r CS_i^r(a_i^r, a_{-i}^r)\Big) \\ & = \arg\max_{a_i} \sum_r CS_i^r(a_i^r, a_{-i}^r). \end{aligned}$$

We can express the cost saving $CS_i^r(a_i^r, a_{-i}^r)$ of node i by substituting (2)

$$\begin{aligned} CS_i^r(a_i^r, a_{-i}^r) \quad = \quad & w_i^r\left[\beta_i(1-\pi_i^r(a_{-i}^r)) + \gamma_i \pi_i^r(a_{-i}^r)\right] \\ & - w_i^r\left(\alpha_i a_i^r + (1-a_i^r)\left[\beta_i(1-\pi_i^r(a_{-i}^r)) + \gamma_i\pi_i^r(a_{-i}^r)\right]\right) \\ = \quad & a_i^r w_i^r\left[\beta_i(1-\pi_i^r(a_{-i}^r)) + \gamma_i\pi_i^r(a_{-i}^r) - \alpha_i\right]. \end{aligned}$$

Thus $CS_i^r(0, a_{-i}^r) = 0$. Then, by defining $\delta_i \triangleq \frac{\beta_i - \alpha_i}{\gamma_i - \alpha_i}$ and $w_i^r[\gamma_i - \alpha_i] = c_{ir}$, we obtain that $CS_i^r(1, a_{-i}^r) = U_i^r(1, a_{-i}^r)$ defined in (1). This model of object replication was used, for example, in [17] to model distributed cache allocation in a computer architecture. It was used to model cooperative caching of contents among Internet Service Providers (ISPs) in [2], where the ISPs are neighbors if they have a peering agreement, and the costs γ_i, β_i, and α_i correspond to the cost of downloading contents over transit, peering and local links, respectively. The model also captures the cost structure of cooperative caching for multiview video streaming systems considered in [19], in which case the costs correspond to the access of image partitions from a remote repository, remote servers and local servers, respectively.

2.2 Graph Multi-coloring, Distributed Radio Spectrum Allocation and Medium Access Control

Proper graph multi-coloring is a generalization of the proper graph coloring problem. Given a graph $\mathcal{G} = (N, E)$ and a set $\mathcal{R}$ of colors the task is to assign K_i distinct colors to vertex $i \in N$ such that no adjacent vertices have the same color. In general, the goal is to use as few as possible colors. In the case of an improper coloring the same color can be assigned to adjacent vertices, but this involves a penalty δ_i for vertex i.

Graph multi-coloring has a number of applications. In the case of scheduling the nodes are jobs, the colors are time units, and K_i is the time needed to finish job i. The minimum number of colors is then the makespan of all jobs [18]. In the case of medium access control there is a set N of nodes that contend for some radio channels or for time slots [3, 4]. The influence graph $\mathcal{G}$ models the potential conflicts due to co-channel interference that can occur between nodes. The model of an undirected influence graph is appropriate for a system in which the pairs of nodes that communicate to each other are relatively close to each other, and fast fading is not a dominant fading component. In the case of channel assignment, each node $i \in N$ has K_i radio interfaces that it can use to transmit data. Alternatively, in the case of time slot allocation, each node $i \in N$ can use K_i time slots. The throughput that node i can achieve transmitting on resource $r \in \mathcal{R}$ in the absence of interference is c_{ir}. In the case of interference the expected throughput drops to $\delta_i c_{ir}$, where δ_i is the resource deterioration expected by node i under interference. In some applications a node can represent an entire network. In the case of coexisting Zigbee PRO networks, for example, the coordinator of each network $i \in N$ is in charge of selecting the channel for the entire network, so as to minimize the expected interference with other networks.

3 Existence of Equilibria and Convergence

We start with addressing two fundamental questions. First, we address the question whether in the graphical resource allocation problem there exist equilibrium resource allocations, from which no player would like to deviate unilaterally. Second, we address the question whether the players could reach an equilibrium state in a distributed manner.

3.1 Existence of equilibria

In order to formalize equilibrium allocations in the resource allocation game and to formulate our results, we start with the definition of three key terms used throughout the section.

Definition 1. A *best reply* of player i is a best strategy a_i^* of player i given the other players' strategies, that is, $U_i(a_i^*, a_{-i}) \geq U_i(a_i, a_{-i}),\ \ \forall\ a_i \in \mathcal{A}_i$.

A *best reply improvement path* is a sequence of strategy profiles, such that in every step t there is one player that *strictly increases* its utility by updating her strategy through performing a *best reply.* Note that a best reply improvement path implies that only one player at a time updates her strategy; this property is known in the literature as asynchronous updates. An improvement path terminates when no player can perform an improvement step. The resulting strategy profile is a pure strategy NE, defined as a strategy profile a^* in which every player's strategy is a best reply to the other players' strategies

$$U_i(a_i^*, a_{-i}^*) \geq U_i(a_i, a_{-i}^*)\ \ \forall\ a_i \in \mathcal{A}_i,\ \forall\ i \in N. \tag{3}$$

Example 1. *Consider* $|N| = 2$ *players,* $|\mathcal{R}| = 3$ *resources and* $K_i = 1$*. Let* $c_{11} > c_{12} > c_{13} > c_{11}\delta_1 > ..$ *and* $c_{22} > c_{22}\delta_2 > c_{21} > c_{23} > ...$ *If the initial strategy profile is* $(3,1)$ *then the following is a best reply improvement path where the unique deviator is marked at every step:* $(3,1) \underset{1}{\rightarrow} (2,1) \underset{2}{\rightarrow} (2,2) \underset{1}{\rightarrow} (1,2)$*. Observe that* $(1,2)$ *is a NE.*

Let us now turn to the question whether all graphical resource allocation games have at least one pure strategy Nash equilibrium. Consider the strategy profile $a(0)$ that consists of the best replies that the players would play on an edgeless social graph. Let us consider now a best reply path starting from the strategy profile $a(0)$. For $t \leq n$ each player $i \in N$ has a chance to play her first best reply at $t = i$. For $t > n$ they play in an arbitrary order. We can make two important observations about the players' best replies.

Lemma 1. *Player* $i \in N$ *allocates only* i-free *resources when she* first *updates her strategy at* $t = i$*.*

Proof. Player i first updates her strategy at $t = i$. Define the evicted set as $E_i(t) = \{r | a_i^r(0) = 1 \wedge a_i^r(t) = 0\}$ and the inserted set as $I_i(t) = \{r | a_i^r(0) = 0 \wedge a_i^r(t) = 1\}$. Consider now two resources $r \in E_i(t = i)$ and $r' \in I_i(t = i)$. By definition $a_i^r(0) = 1$ and $a_i^{r'}(0) = 0$, thus by the definition of best reply and because of the edgeless social graph at $t = 0$

$$c_{ir} \geq c_{ir'} \tag{4}$$

Since r' is allocated and r is evicted, at the improvement step $t = i$ it must hold that

$$U_i^{r'}(1, a_{-i}^{r'}(i)) > U_i^r(1, a_{-i}^r(i)). \tag{5}$$

Consider now the definition of the payoff (1) and (4). If r is *i-free* then $U_i^r(1, a_{-i}^r(i)) = c_{ir}$ and consequently (5) cannot hold. Similarly, if r' is *i-busy* then $U_i^{r'}(1, a_{-i}^{r'}(i)) = \delta_i c_{ir'}$ and consequently (5) cannot hold. Thus the only possibility to satisfy (5) is that r' is *i-free* and r is *i-busy.* $\square$

Lemma 2. *Consider a sequence of best reply steps and the best reply $a_i(t)$ played by player i at step $t > 0$. A necessary condition for $a_i(t)$ not being a best reply for i at step $t' > t$ is that at least one of the following holds*

*(i) An i-*free* resource r allocated by i ($a_i^r(t) = 1$, $\pi_i^r(t) = 1$) becomes i-*busy* by step t',*

*(ii) An i-*busy* resource r not allocated by i ($a_i^r(t) = 0$, $\pi_i^r(t) = 0$) becomes i-*free* by step t'.*

The proof of Lemma 2 is in the Appendix. Lemma 1 implies that a resource allocated by player i cannot change from *i-free* to *i-busy* during the first round of best reply steps ($1 \leq t \leq n$). Consequently, condition (i) in Lemma 2 cannot hold, and the only reason why player i would update her strategy a second time (at some $t > n$) is that condition (ii) holds, and she would start allocating an *i-free* resource. Thus, by induction, condition (i) will never hold, and in every step t player $i \in N$ starts allocating only *i-free* resources. Since no player ever starts allocating an *i-busy* resource, according to the expression of the payoff in (1) the utilities of the players cannot decrease for $t > 0$. Nevertheless, every time a player updates her strategy her utility must strictly increase. Since the players' utilities cannot increase indefinitely, the best reply path must end in a NE. Hence we can state the following.

Theorem 1. *Every graphical resource allocation game possesses a pure strategy Nash equilibrium.*

For a complete influence graph every player makes at most one best-reply improvement step [20], but this does not hold even for a simple non-complete influence graph: on a ring of 4 players with $c_{ir} = c_{jr}$ ($\forall r \in \mathcal{R}$) at least one player updates her strategy twice. We can however bound the number of required improvement steps. We know that from $a(0)$ every player can only start allocating *i-free* resources. In the worst case each player i inserts exactly one *i-free* resource r at every best reply step. According to condition (ii) in Lemma 2, r becomes *i-free* as an effect of a best reply of a player $j \in \mathcal{N}(i)$. The number of resources that can change from *i-busy* to *i-free* for an arbitrary player i are at most $\sum_{j \in \mathcal{N}(i)} K_j$, thus we obtain the following result

Theorem 2. *It is possible to compute a Nash equilibrium of a graphical resource allocation game in at most $\sum_{i \in N} \sum_{j \in \mathcal{N}(i)} K_j$ steps.*

Due to space limitations we do not address issues such as the price of anarchy and price of stability, and the occurrence of phenomena such as Bélády's anomaly [21]. Instead, in the rest of the paper our focus is on how to reach Nash equilibria in an efficient way, with the aim of providing guidelines for designing distributed algorithms that would be able to reach an equilibrium allocation with little overhead.

3.2 Convergence to Nash Equilibria

The existence of equilibrium states is important, but in a distributed system it is equally important to have efficient distributed algorithms that the nodes can use to reach an equilibrium state. The algorithm in Theorem 1 can be used to reach an equilibrium state if the values c_{ir} of the resources in the nodes never change, so once a NE is reached, the nodes will not deviate from it. Nevertheless, the algorithm would be inefficient if the values c_{ir} or the influence graph can change over time, as the equilibrium states for different values and different influence graphs are, in general, different. Hence, an important question is whether the players will reach a NE given an arbitrary initial strategy profile, e.g., a NE for past c_{ir} or a NE for a past influence graph, and given the myopic decisions the players make to update their strategies.

A straightforward distributed algorithm would be to let every player play her best reply at the same time (synchronously) until an equilibrium is reached. The algorithm only requires synchronization between neighboring players, thus it is relatively easy to implement. Unfortunately, it is easy to find resource allocation games where players cannot find an equilibrium this way.

Example 2. *Consider two players and $|\mathcal{R}| \geq 2$. Let $c_{i1} > c_{i2} > c_{i1}\delta_i > c_{i2}\delta_i$ and $K_i = 1$. If the initial allocation strategies are $a_i(0) = (1,0)$ then we have $(1,0) \underset{1,2}{\longrightarrow} (0,1) \underset{1,2}{\longrightarrow} (1,0)$, etc.*

As an alternative, consider an algorithm that only allows one player to play a best reply at a time. This is the case of the asynchronous updates used in Section 3.1, which generate best reply improvement paths. The algorithm requires global synchronization to ensure that no players perform an update simultaneously. In what follows we show that even best reply improvement paths can be arbitrarily long.

Consider a resource allocation game played over the influence graph shown in Figure 1, where $\mathcal{R} = \{a, b, c, d\}$, and $K_i = 1$. Each player $1 \leq i \leq 4$ has a resource $r^* \in \mathcal{R}$ such that $c_{ir^*}\delta_i > c_{ir}\delta_i \ \forall r \neq r^*$, and at least one resource $r' \in \mathcal{R}$ such that $c_{ir'} > c_{ir^*}\delta_i$. For players $5 \leq i \leq 8$ there is a resource $r^* \in \mathcal{R}$ such that $c_{ir^*}\delta_i > c_{ir} \ \forall r \neq r^*$. In the following we show an asynchronous best reply dynamic that cycles, we omit the strategies of players $5 \leq i \leq 8$ since they always allocate the resource that has the highest value at their respective neighboring player $i-4$. The *i-busy* resources are in bold.

$$
\begin{aligned}
(\boldsymbol{a}, b, \boldsymbol{d}, \boldsymbol{a}) \underset{3}{\rightarrow} \quad (\boldsymbol{a}, b, c, \boldsymbol{a}) \underset{1}{\rightarrow} \quad (\boldsymbol{b}, \boldsymbol{b}, c, \boldsymbol{a}) \underset{4}{\rightarrow} \quad (\boldsymbol{b}, \boldsymbol{b}, c, d) \underset{2}{\rightarrow} \quad (\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{c}, d) \\
\underset{1}{\rightarrow} \quad (a, \boldsymbol{c}, \boldsymbol{c}, d) \underset{3}{\rightarrow} \quad (a, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{d}) \underset{2}{\rightarrow} \quad (a, b, \boldsymbol{d}, \boldsymbol{d}) \underset{4}{\rightarrow} \quad (\boldsymbol{a}, b, \boldsymbol{d}, \boldsymbol{a})
\end{aligned}
$$

The existence of a cycle in the best reply paths implies that resource allocation games played over an arbitrary graph do not admit a potential function [13]. It also raises the question whether there could be an initial strategy from which *every*

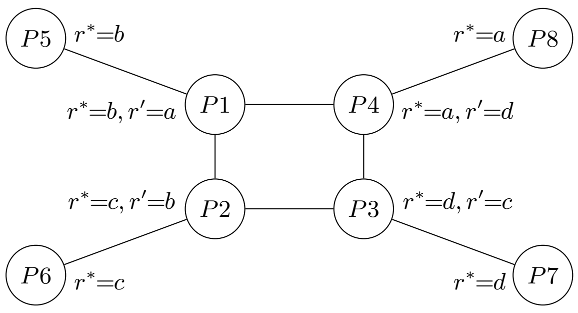


Figure 1: Influence graph and players' preferences that allow a cycle in best replies.

best reply improvement path is infinite. Starting a best reply improvement path from such an initial strategy profile, the players would never reach a NE using the asynchronous algorithm.

We show that fortunately this is not the case; from every strategy profile there exists at least one finite best reply path that leads to a NE. This property is called weak acyclicity [14, 22].

Definition 2. A game is *weakly acyclic under best replies* if from every strategy profile a, there is a best reply improvement path starting from a and ending in a pure Nash equilibrium.

To show that resource allocation games are weakly acyclic, let us consider a best reply $a_i(t)$ of a player i at step t. We can define four not mutually exclusive properties of $a_i(t)$, depending on whether the involved resources are *i-busy* as shown in Table 1.

1	$\exists r \in E_i(t)$, r *i-busy*	$\exists r' \in I_i(t)$, r' *i-busy*
2	$\exists r \in E_i(t)$, r *i-busy*	$\exists r' \in I_i(t)$, r' *i-free*
3	$\exists r \in E_i(t)$, r *i-free*	$\exists r' \in I_i(t)$, r' *i-busy*
4	$\exists r \in E_i(t)$, r *i-free*	$\exists r' \in I_i(t)$, r' *i-free*

Table 1: Four properties of a best reply move

Note that a best reply necessarily has at least one of the listed properties. The following two lemmas state that the best replies in a cycle cannot be arbitrary.

Lemma 3. *In every best reply cycle there exists at least one strategy profile $a(t)$ in which at least one player $i \in N$ performs a best reply that has property 1). Furthermore, in a best reply cycle it is not possible to perform a best reply that has property 3).*

Lemma 4. *Assume that a player i performs a best reply with property 1) at step t in a best reply cycle such that $r' \in I_i(t)$ and r' is i-*busy. *Then every resource $r'' \in \mathcal{R}$ for which $c_{ir''} > c_{ir'}$ is allocated by player i at step t ($a_i^{r''}(t)=1$).*

The proofs of the lemmas are given in the Appendix. Let us consider a strategy profile $a(t)$ in a best reply cycle in which at least one player can perform a best reply that has property 1). Such a strategy profile exists according to Lemma 3. Starting from $a(t)$, let us perform a sequence of best replies with property 1). From Lemma 4, it follows that player i cannot evict the resources that she inserted as *i-busy* through these best replies, thus every player i can perform at most K_i best replies with property 1). So after at most $\sum_{i \in N} K_i$ best replies with property 1) we reach a strategy profile $a(t')$ in which there is no player that can perform a best reply that has property 1). If in $a(t')$ no player can make a best reply then $a(t')$ is a NE. Otherwise, if a player i can perform a best reply in $a(t')$ then it will have neither property 1) nor 3). As a consequence of a best reply that has property 2) or 4) only, condition (i) of Lemma 2 cannot hold for any player, so the only new reason why a player j would perform a best reply is that condition (ii) in Lemma 2 is satisfied, and she would start allocating *j-free* resources. Thus, by induction, condition (i) of Lemma 2 will never hold, and in every step $t'' > t'$ players only perform best replies that have property 2) or 4) but not 1) or 3). A player's utility strictly increases when it performs a best reply, and a best reply with property 2) or 4) does not decrease any other player's utility. Since the players' utilities cannot increase indefinitely, this path must end in a NE after a finite number of steps. Hence we can state the following.

Theorem 3. *Every graphical resource allocation game is weakly acyclic under best replies.*

Weak acyclicity has two important consequences for system design. First, if the nodes that compete for the resources update their allocations one at a time in a myopic way and the order of updates is fixed then the nodes might cycle forever without reaching an equilibrium. Therefore, it is advisable that the next node to perform the update step is chosen at random by the system. By doing so the system will reach an equilibrium after a finite number of update steps *on average*, even though the number of update steps can be arbitrarily high. Second, weak acyclicity in best replies implies that various complex learning rules can be used to reach an equilibrium with high probability. One example is adaptive play, described in [22], when nodes select the next allocation simultaneously based on a random sample of a finite history of the allocations. Such a learning rule resembles a system in which the nodes update their allocations based on a sliding window estimator of the other players' allocations. Another example is regret based learning [23], in which case every node updates its allocation simultaneously so as to minimize its loss of utility in retrospect given the history of all allocations.
In the discussion above we showed that every best reply path that starts from a

strategy profile $a(t')$ in which no player can perform a best reply that has property 1) ends in a NE. It follows that in every strategy profile of a best reply cycle there is at least one player i that can perform a best reply with property 1). Let us consider now the number of steps needed to reach a NE starting from an arbitrary strategy profile $a(t)$. We have already shown that after performing at most $\sum_{i \in N} K_i$ best replies with property 1) we reach a strategy profile $a(t')$ in which there is no player that can perform a best reply that has property 1). Given that there is no player that can perform a best reply with property 1), we can use the same reasoning that we formulated to show Theorem 2 to bound the maximum number of best reply steps from $a(t')$ to a NE to $\sum_{i \in N} \sum_{j \in \mathcal{N}(i)} K_j$. Summing it to the maximum number of steps required from $a(t)$ to $a(t')$ we obtain the following upper bound.

Corollary 1. *From an arbitrary strategy profile there exists a best reply path that reaches a NE in at most* $\sum_{i \in N} K_i + \sum_{i \in N} \sum_{j \in \mathcal{N}(i)} K_j$ *steps.*

As an example, consider a graphical resource allocation game with $K_i = K$ $\forall i \in N$ and a regular influence graph of degree d. In this case the maximum length of a best reply path would be $|N| \cdot K + |N| \cdot d \cdot K = |N|K(1 + d)$.

4 The Case of Complete Influence Graph

In this section we consider the case that the influence graph is *complete* and use it to illustrate the influence of the graph topology on the convergence to equilibria. We make use of the notion of the finite best reply property [14].

Definition 3. A game possesses the *finite best reply property* (FBRP) if every best reply improvement path is finite.

For a complete influence graph the resource allocation game is a special case of the player-specific matroid congestion games investigated in [16], and it is known that if $|\mathcal{A}_i| = 2$ $\forall i \in N$ or if $|N| = 2$, then the game has the FBRP. It is not known whether the game possesses the FBRP in general.

Let us denote by $F_i(t)$ the set of *i-free* resources and by $B_i(t)$ the set of *i-busy* resources allocated by i at step t. Note that $|F_i(t)| + |B_i(t)| = K_i$. We can prove the following lemmas.

Lemma 5. *On a complete influence graph, a best reply step $a_i(t)$ performed by player i in a best reply cycle cannot affect the number of i-*busy *resources allocated by i, that is, $|B_i(t)| = |B_i(t-1)|$.*

The proof of Lemma 5 is given in the Appendix.

Lemma 6. *On a complete influence graph, for a best reply step $a_i(t)$ performed by player i in a best reply cycle it holds that $c_{ir'} < c_{ir}$, $\forall r' \in E_i(t), r \in I_i(t)$.*

Proof. Player i solves a knapsack problem to construct her best reply. Recall that according to Lemma 5 we have $|B_i(t-1)| = |B_i(t)|$ and consequently $|F_i(t-1)| = |F_i(t)|$. Hence we can construct the best reply of player i by dividing the knapsack problem into two similar subproblems: we can solve the knapsack problem for all the *i-busy* resources and populate the set $B_i(t)$, and do the same with the set $F_i(t)$ using the *i-free* ones. Suppose that k resources are evicted from one set, consequently k are inserted, and in order for the result to be the solution of the knapsack problem, the cost saving yielded by each inserted resource must be higher than the cost saving yielded by each evicted resource. That is, for every $r, r' \in \mathcal{R}$ such that r was inserted and r' was evicted from $B_i(t)$, we have $c_{ir}\delta_i > c_{ir'}\delta_i \Rightarrow c_{ir} > c_{ir'}$. Similarly, for every $r, r' \in \mathcal{R}$ such that r was inserted and r' was evicted from $F_i(t)$, we have $c_{ir} > c_{ir'}$. This proves the lemma. □

Assume now that there is a best reply cycle. According to Lemma 6 each best reply step can only move towards resources with higher value. The number of resources $|\mathcal{R}|$ is finite, hence every player can only perform a finite number of best replies and the best reply path terminates after a finite number of steps. Thus the following result holds.

Theorem 4. *Every resource allocation game played over a complete influence graph has the FBRP.*

Theorem 4 has important practical implications. Recall that based on the results for a general graph topology shown in Section 3.2 it was advisable to ensure that nodes would update their allocations one at a time and in a random order so as to ensure that an equilibrium allocation can be reached; the number of update steps was unbounded in general but finite on average. By Theorem 4, if the influence graph is complete and the nodes that compete for the resources update their allocations one at a time by using some synchronization protocol, then they will reach an equilibrium allocation after a *bounded* number of update steps starting from an arbitrary initial allocation independent of the order of the updates. Since the order of updates can be arbitrary (e.g,. fixed), the system would require less coordination between the nodes.

5 The Importance of the Utility Structure

In this section we allow an arbitrary influence graph but we introduce a constraint on the payoff structure: we consider the case when $\delta_i = 0$. This case was used in the context of replication to model the problem of cooperative caching between peering ISPs in [2], and in the context of radio spectrum allocation $\delta_i = 0$ corresponds to the case when interference leads to zero expected throughput (i.e., proper multi-coloring). Throughout the section we consider a notion of improvement paths that we refer to as lazy.

Definition 4. A step $a_i(t+1)$ of player i is an *improvement step* if $U_i(a_i(t+1), a_{-i}(t)) > U_i(a_i(t), a_{-i}(t))$. A *lazy improvement step* of player i is an improvement step such that the payoff of every inserted resource exceeds that of every evicted resource. That is, for every $r \in E_i(t+1)$ and $r' \in I_i(t+1)$ we have $U_i^r(1, a_{-i}^r(t)) < U_i^{r'}(1, a_{-i}^{r'}(t))$.

Clearly, every best reply improvement step is a lazy improvement step. The following result shows that all lazy improvement paths are finite for arbitrary graph topologies if $\delta_i = 0$.

Proposition 5. *In a graphical resource allocation game with $\delta_i = 0$ $\forall i \in N$ every lazy improvement path is finite, i.e., the game possesses the* finite lazy improvement property*.*

Proof. We prove the proposition by showing that under the condition $\delta_i = 0$ the resource allocation game has a generalized ordinal potential function [13] for lazy improvement steps. A function $\Psi : \times_i(\mathcal{A}_i) \to \mathbb{R}$ is a generalized ordinal potential function for the game if the change of Ψ is strictly positive if an arbitrary player i increases its utility by changing her strategy from a_i to a_i'. Formally, $U_i(a_i, a_{-i}) - U_i(a_i', a_{-i}) > 0 \Rightarrow \Psi(a_i, a_{-i}) - \Psi(a_i', a_{-i}) > 0$. In the following we show that the function

$$\Psi(\mathbf{a}) = \sum_i U_i(a_i, a_{-i}), \tag{6}$$

where the utility function was defined in (1), is a generalized ordinal potential for the game. Substituting $\delta_i = 0$ into (1), one notes that player i benefits only from allocating *i-free* resources. Furthermore, the utility of player i does not depend on resources that she does not allocate herself. Given a strategy profile $\mathbf{a} = (a_i, a_{-i})$ player i can improve its utility by combining three kinds of lazy improvement steps.

First, if player i has free storage capacity (that is, $\sum_r a_i^r < K_i$) then she has to allocate a resource r for which $a_i^r = 0$ but $U_i^r(1, a_{-i}^r) > 0$. By (1) we know that resource r is *i-free* and hence the utility of her neighbors will not be affected if she allocates resource r. Consequently, if we denote the new strategy of player i by $a_i' = (a_i^1, \dots, a_i^{r-1}, 1, a_i^{r+1}, \dots, a_i^{|\mathcal{R}|})$ then

$$\Psi(a_i', a_{-i}) - \Psi(a_i, a_{-i}) = U_i(a_i', a_{-i}) - U_i(a_i, a_{-i}) = U_i^r(1, a_{-i}^r) > 0. \tag{7}$$

Second, if player i stops allocating a resource r for which $U_i^r(1, a_{-i}^r) = 0$ and starts allocating a resource r' for which $U_i^{r'}(1, a_{-i}^{r'}) > 0$. By (1) we know that resource r is *i-busy*, but resource r' is *i-free*. Let us denote the strategy of player i after the change by a_i'. We first observe that the utility of the neighboring players cannot decrease when player i stops allocating resource r (it can potentially increase). At the same time the utility of the neighboring players does not change when player i starts allocating resource r'. Hence we have that

$$\Psi(a_i', a_{-i}) - \Psi(a_i, a_{-i}) \geq U_i^{r'}(1, a_{-i}^{r'}) > 0. \tag{8}$$

Third, if player i stops allocating a resource r for which $U_i^r(1, a_{-i}^r) > 0$ and starts allocating a resource r' for which $U_i^{r'}(1, a_{-i}^{r'}) > U_i^r(1, a_{-i}^r)$. By (1) we know that both resource r and r' are *i-free*. Hence the utility of the neighboring players is not affected by the change. The utility of player i increases, however. Let us denote the strategy of player i after the change by a_i', then

$$\Psi(a_i', a_{-i}) - \Psi(a_i, a_{-i}) = U_i^{r'}(1, a_{-i}^{r'}) - U_i^r(1, a_{-i}^r) > 0. \tag{9}$$

By summing (7)-(9) we can see that the function Ψ is a generalized ordinal potential function for an arbitrary combination of lazy improvement steps. Finite games that allow a generalized ordinal potential function were shown to have the finite improvement property in [13]. Following the arguments of (Lemma 2.3, [13]) it follows that the resource allocation game has the finite lazy improvement property. We note that if non-lazy improvement steps are allowed, then a player can allocate an *i-busy* resource as part of an improvement step and thus the proof would not hold. $\square$

The finite lazy improvement property shown in Proposition 5 allows more freedom for system design than the results in Theorems 3 and 4 that are valid for best reply updates. Based on Proposition 5 an update step can be as simple as evicting the worst allocated resource and inserting the best non-allocated resource, and the system is guaranteed to reach an equilibrium for an arbitrary graph topology in a bounded number of steps.

6 Fast convergence based on independent sets

Until now we considered asynchronous updates. Unfortunately, the implementation of the asynchronous update rule in a distributed system would require global synchronization, which can be impractical in large distributed systems. Hence, an important question is whether the previous results of convergence to a NE would be preserved even if some players would update their strategies simultaneously. In the following we show that relaxing the requirement of asynchronicity is indeed possible: both Theorem 3 and Proposition 5 hold if the players that simultaneously make an update at each step form an independent set of the influence graph $\mathcal{G}$, that is, two players i,$j \in N$ can make an update simultaneously only if $j \notin \mathcal{N}(i)$. We refer to this dynamic as the *plesiochronous* dynamic.

Proposition 6. *Every resource allocation game is weakly acyclic under plesiochronous best replies.*

The proof of the proposition is analogous to the proof of Theorem 3.

Proposition 7. *In a graphical resource allocation game with $\delta_i = 0\ \forall i \in N$ every plesiochronous lazy improvement path is finite.*

Proof. Consider a sequence of the subsets of the players $N^*(t) \subseteq N$ $(t = 0, \ldots)$ such that for $i, j \in N^*(t)$ we have $j \notin \mathcal{N}(i)$. Observe that, by definition, each set $N^*(t)$ is an independent set of the influence graph $\mathcal{G}$. The players $i \in N^*(t)$ make an improvement step at step t simultaneously from $a_i(t-1)$ to $a_i(t)$. Each player can combine the three kinds of lazy improvement steps discussed in the proof of Proposition 5 to increase its utility.

Since we require that none of the players that update their strategies are neighbors of each other, then their updates do not affect each others' utilities. Formally, for $i \in N^*(t)$ we have $U_i(a_i(t), a_{-i}(t)) = U_i(a_i(t), a_{-i}(t-1))$. Consequently, we can use the same arguments as in the proof of Proposition 5 to show for every $i \in N^*(t)$ that Ψ defined in (6) satisfies $U_i(a_i(t), a_{-i}(t)) - U_i(a_i'(t-1), a_{-i}(t-1)) > 0 \Rightarrow \Psi(a(t)) - \Psi(a(t-1)) > 0$.
The rest of the proof is similar to that of Proposition 5. $\square$

In order to maximize the convergence speed of the plesiochronous dynamic we need to find a minimum vertex coloring of $\mathcal{G}$, i.e., we have to find the chromatic number $\chi(\mathcal{G})$ of graph $\mathcal{G}$. Finding the chromatic number is NP-hard in general, but efficient distributed graph coloring algorithms exist [24], and can be used to find a coloring in a distributed system. The chromatic number can be bounded based on the largest eigenvalue $\lambda_{max}(\mathcal{G})$ of the graph's adjacency matrix [25],

$$\chi(\mathcal{G}) \leq 1 + \lambda_{max}(\mathcal{G}) \leq \max_{i \in N} |\mathcal{N}(i)|$$

where the second inequality follows from the Perron-Frobenius theorem ([26], Lemma 2.8). Given a coloring, the number of steps needed to reach equilibrium can be significantly smaller than for the corresponding asynchronous dynamic for sparse graphs.

We illustrate the convergence speedup of the plesiochronous dynamic compared to the asynchronous dynamic in Figures 2 and 3. For the plesiochronous dynamic we used the Welsh-Powell algorithm to find a coloring [27] of the influence graphs and we denote the number of colors by $\xi(\mathcal{G})$. Each player can allocate $K_i = 5$ resources and we considered two scenarios: $\delta_i = 0$ and $\delta_i = 0.5$. Figure 2 shows the average number of lazy improvement steps needed to reach equilibrium as a function of the average node degree in Erdős-Rényi random graphs with 87 vertices. Each data point is the average of the results obtained on 160 random graphs with the same average node degree. The figure shows the 95% confidence intervals for the case $\delta_i = 0$. We omitted the confidence intervals for $\delta_i = 0.5$ to improve readability. The results confirm that the plesiochronous dynamic converges significantly faster than the asynchronous dynamic, especially over sparse influence graphs. The figure also confirms that the convergence properties are different on a complete influence graph than on a sparse graph, as the number of steps necessary for the asynchronous dynamic to reach a NE drops for average node degree equal to 86.
Figure 3 shows the speedup (the ratio of the number of steps needed to reach equilibrium under the asynchronous dynamic and the plesiochronous dynamic) as a

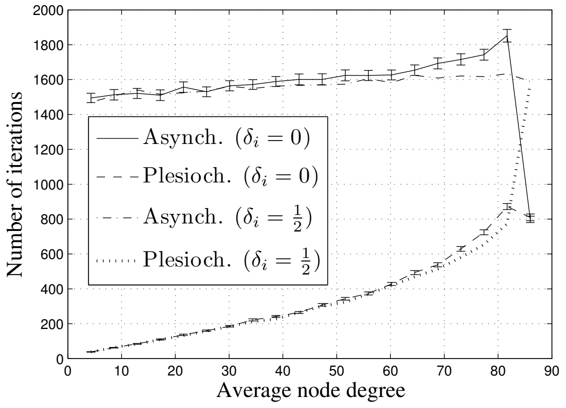


Figure 2: Average number of improvement steps needed to reach a NE for asynchronous and plesiochronous dynamics vs. the average node degree of the ER random graphs.

function of the average node degree for Erdős-Rényi (ER) and for Barabási-Albert (BA) random graphs with 87 vertices for $\delta_i = \frac{1}{2}$. Each data point is the average of the results obtained on 160 random graphs with the same average node degree. The results show that the speedup for BA random graphs is slightly smaller than for ER random graphs but shows a similar trend. A comparison of the speedup with the average size of the independent sets after the coloring of the random graphs ($\frac{|N|}{\xi(\mathcal{G})}$) indicates that the speedup of the plesiochronous dynamic exceeds the average number of players that can perform an improvement step simultaneously.

The results formulated in Propositions 6 and 7 allow one to simplify the design of large systems significantly. The convergence results of Sections 3, 4 and 5 required that only one node at a time would update its allocation. In order to ensure this, there needs to be a coordination protocol for the entire system that ensures that there be only one node at a time that updates its allocation. In contrast, Propositions 6 and 7 show that local coordination is sufficient to ensure that the system would reach an equilibrium allocation, and the numerical results show that convergence is in fact very fast. In practice the information needed for the coordination can be piggybacked with the information about the updated allocations, and thus an equilibrium allocation can be reached with very little communication overhead.

7 Related Work

There is a large body of works on congestion games [5] on complete influence graphs. Most works consider congestion games that allow a potential function [13], and analyze the number of steps needed to reach equilibria [9], the price of anarchy and the price of stability [10, 11], or the complexity of calculating equilibria [12].

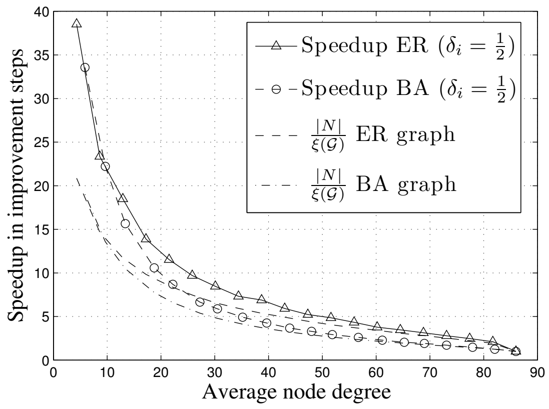


Figure 3: Speedup in terms of the number of improvement steps needed to reach a NE vs. the average node degree of the ER and the BA random graphs.

Player-specific congestion games on a complete influence graph that do not admit a potential function were considered in [14, 20, 15, 16]. In [14] it was shown that for non-weighted player-specific congestion games with *singleton action sets* the best reply paths are finite. In [20] the authors showed the existence of equilibria for a game of replication. In [15], the authors considered congestion games with player-specific constants, which correspond to all players having the same cost of sharing($\delta_i = \delta$) in our model, and showed that improvement paths are finite in the unweighted version of the game. In [16] the authors showed that player-specific congestion games with matroid action sets are weakly acyclic in better replies, and provided bounds on the length of the shortest improvement paths. They also showed that games with 2 players, or with 2 actions per player are acyclic in best replies, similar to [14].

A few recent works considered graphical congestion games that allow potential functions [6, 7, 8]. [6] gave results on the price of anarchy and stability for games with linear payoff functions, and showed that cycles can exist if the influence graph is directed and contains cycles. [7] addressed similar questions for weighted graphical congestion games with linear payoff functions. In [8] the authors analyzed the number of steps needed to reach equilibrium in graphical congestion games with homogeneous resources and singleton action sets.

The resource allocation game we consider combines the concept of graphical congestion games with player-specific payoffs on non-singleton action sets. Our results on equilibrium existence, and the results on convergence rely on non-standard techniques and could be of interest for the analysis of congestion games that do not admit a potential function. Furthermore, the proposed plesiochronous update dynamic based on independent sets is a promising candidate for implementation in large distributed systems.

8 Conclusion

In this work we considered a player-specific graphical resource allocation game, a resource allocation game played over an influence graph. The game models a system in which every node can choose a subset of resources, and the value of a selected resource to a node is decreased if any of its neighbors chooses the same resource. We showed that pure strategy Nash equilibria exist in the game for arbitrary influence graph topologies even though the game does not admit a potential function, and gave a bound on the complexity of finding equilibria. We then considered the problem of reaching an equilibrium when nodes update their allocations one at a time to their best allocation, that is, every node performs a best reply. We showed that for non-complete influence graphs there might be cycles in best replies but the system would reach an equilibrium eventually if updates are done in a random order. For complete influence graphs there are no cycles, and hence an equilibrium is reached after a finite number of steps. We also showed that if neighboring nodes try to allocate disjoint sets of resources then there are no cycles, even if the nodes' update steps are improvements instead of best replies. Finally, we showed that the convergence properties hold even if improvement steps are performed simultaneously by nodes that form an independent set of the influence graph, and proposed an efficient algorithm to reach a NE over sparse influence graphs that only requires local coordination between neighboring nodes.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of ACM CoNEXT*, 2009.

[2] G. Dán, "Cache-to-cache: Could ISPs cooperate to decrease peer-to-peer content distribution costs?" *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1469–1482, 2011.

[3] L. Narayanan, "Channel assignment and graph multicoloring," in *Handbook of wireless networks and mobile computing*, 2002, pp. 71–94.

[4] K. I. Aardal, V. Hoesel, P. M. Stan, A. M. C. A. Koster, C. Mannino, and A. Sassano, "Models and solution techniques for frequency assignment problems," *Annals of Operations Research*, vol. 153, no. 1, pp. 79–129, 2007.

[5] R. W. Rosenthal, "A class of games possessing pure-strategy Nash equilibria," *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.

[6] V. Bilò, A. Fanelli, M. Flammini, and L. Moscardelli, "Graphical Congestion Games." in *Proc. of WINE*, vol. 5385, 2008, pp. 70–81.

[7] D. Fotakis, V. Gkatzelis, A. C. Kaporis, and P. G. Spirakis, “The Impact of Social Ignorance on Weighted Congestion Games,” in *Proc. of WINE*, 2009, pp. 316–327.

[8] R. Southwell and J. Huang, “Convergence Dynamics of Resource-Homogeneous Congestion Games,” in *Proc. of GameNets*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 75, no. 412509, 2011, pp. 281–293.

[9] S. Chien and A. Sinclair, “Convergence to approximate Nash equilibria in congestion games,” in *Proc. of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 169–178.

[10] G. Christodoulou and E. Koutsoupias, “The price of anarchy of finite congestion games,” in *Proc. of ACM Symposium on Theory of Computing*, 2005, pp. 67–73.

[11] B. Awerbuch, Y. Azar, and A. Epstein, “The price of routing unsplittable flow,” in *Proc. of ACM Symposium on Theory of Computing*, 2005, pp. 57–66.

[12] A. Fabrikant, C. Papadimitriou, and K. Talwar, “The complexity of pure Nash equilibria,” in *Proc. of ACM Symposium on Theory of Computing*, 2004, pp. 604–612.

[13] D. Monderer and L. S. Shapley, “Potential games,” *Games and Economic Behavior*, vol. 14, pp. 124–143, 1996.

[14] I. Milchtaich, “Congestion games with player-specific payoff functions,” *Games and Economic Behavior*, vol. 13, no. 1, pp. 111–124, 1996.

[15] M. Mavronicolas, I. Milchtaich, B. Monien, and K. Tiemann, “Congestion games with player-specific constants,” *Mathematical Foundations of Computer Science 2007*, pp. 633–644, 2007.

[16] H. Ackermann, H. Röglin, and B. Vöcking, “Pure Nash equilibria in player-specific and weighted congestion games,” *Theor. Computer Science*, vol. 410, no. 17, pp. 1552–1563, 2009.

[17] A. Leff, J. L. Wolf, and P. S. Yu, “Replication Algorithms in a Remote Caching Architecture.” *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 11, pp. 1185–1204, 1993.

[18] M. M. Halldórsson and G. Kortsarz, “Tools for Multicoloring with Applications to Planar Graphs and Partial k-Trees,” *Journal of Algorithms*, vol. 42, no. 2, pp. 334–366, 2002.

[19] H. Huang, B. Zhang, G. Chan, G. Cheung, and P. Frossard, “Coding and Replication Co-Design for Interactive Multiview Video Streaming,” in *Proc. of mini-conference in IEEE INFOCOM*, 2012, pp. 1–5.

[20] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed selfish replication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 12, pp. 1401–1413, 2006.

[21] L. A. Belady, R. A. Nelson, and G. S. Shedler, "An Anomaly in Space-Time Characteristics of Certain Programs Running in a Paging Machine," in *Communications of the ACM*, vol. 12, no. 6, 1969, pp. 349–353.

[22] H. P. Young, "The evolution of conventions," *Econometrica: Journal of the Econometric Society*, vol. 61, no. 1, pp. 57–84, 1993.

[23] J. R. Marden, G. Arslan, and J. S. Shamma, "Regret Based Dynamics: Convergence in Weakly Acyclic Games," in *Proc. of AAMAS*, 2007, pp. 194–201.

[24] F. Kuhn, "Weak Graph Colorings: Distributed Algorithms and Applications," in *ACM SPAA*, 2009, pp. 138–144.

[25] H. S. Wilf, "The eigenvalues of a graph and its chromatic number," *J. London Math. Soc.*, vol. 42, pp. 330–332, 1967.

[26] R. S. Varga, *Matrix Iterative Analysis*, 2002.

[27] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967.

Proof of Lemma 2. According to the structure of the utility function $U_i(a_i, a_{-i}) = \sum_{\{r|a_i^r=1\}} U_i^r(1, a_{-i})$, a best reply $a_i(t)$ can stop to be such in two situations:

(i) The payoffs of one or more allocated resources $\{r \in \mathcal{R}|a_i^r(t) = 1\}$ decrease;

(ii) The payoffs of one or more not allocated resources $\{r \in \mathcal{R}|a_i^r(t) = 0\}$ increase;

According to the definition of cost saving in (1), case (i) can happen only if some *i-free* resources allocated by i become *i-busy.* This requires that some player $j \in \mathcal{N}(i)$ starts allocating some *j-busy* resources. Similarly, case (ii) can happen only if a neighbor $j \in \mathcal{N}(i)$ allocating an resource r evicts it, making resource r *i-free.* $\square$

Proof of Lemma 3. We prove the lemma by showing that a best reply satisfying condition 1) must occur in a best reply cycle in order for the cycle to exist. The utility of at least one player must decrease at least once in a best reply cycle. According to the definition of cost saving in (1), in order for the utility $U_j(a(t))$ of a player j to decrease, some neighbor $i \in \mathcal{N}(j)$ needs to start allocating some *i-busy* resource allocated by j. It follows that, from Table 1, in a best reply cycle there has to be at least one best reply satisfying condition 1) or 3). Let $r \in E_i(t)$ and $r' \in I_i(t)$ be two resources in the evicted and inserted sets, respectively, during a best reply of player i at step t in a best reply cycle. It follows from the definition of inserted and evicted set that $a_i^r(t-1) = 1$ and $a_i^{r'}(t-1) = 0$. Assume now that this best reply satisfies 3). This implies $c_{ir} < c_{ir'}\delta_i$. Note that this inequality also implies that player i always prefers r' over r (r' always provides a higher cost saving) and thus if $a_i^r(t-1) = 1$, then $a_i^{r'}(t-1) = 1$. Since there cannot be a best reply satisfying 3), there must be at least one best reply satisfying 1). This proves the lemma. □

Proof of Lemma 4. Since r' is *i-busy* it yields to player i the payoff $c_{ir'}\delta_i$. Consider an object r'' for which $c_{ir''} > c_{ir'}$. Since player i is performing a best reply, if r'' is *i-free* then its payoff is $c_{ir''} > c_{ir'}\delta_i$ and consequently $a_i^{r''}(t) = 1$. Similarly, if r'' is *i-busy* then its payoff is $c_{ir''}\delta_i > c_{ir'}\delta_i$, and consequently $a_i^{r''}(t) = 1$. □

Proof of Lemma 5. Part A: First we show that $|B_i(t-1)| \geq |B_i(t)|$. Player i can only increase the number of *i-busy* allocated resources if she evicts at least one *i-free* resource r' from $a_i(t-1)$ and inserts an *i-busy* resource r at step t. Thus by (1) we have

$$c_{ir'} < c_{ir}\delta_i \tag{10}$$

Since we are in a best reply cycle, at some step $t' > t$ the strategy $a_i(t-1)$ must become a best reply for player i, i.e. $a_i(t') = a_i(t-1)$. This requires either $c_{ir'} > c_{ir}$ or $c_{ir'} > c_{ir}\delta_i$, and both contradict (10).

Part B: Second we show that for every step in a best reply cycle $|B_i(t-1)| = |B_i(t)|$ must hold. We denote by $C(t)$ the set of the chosen resources, the resources allocated by at least one player in $a(t)$, $C(t) = \{r | a_j^r(t) = 1 \text{ for some } j \in N\ \} \subseteq \mathcal{R}$. Similarly, we denote by $C(t)_{-i}$ the set of resources allocated by the players not including i, $C(t)_{-i} = \{r | a_j^r(t) = 1 \text{ for some } j \in N \setminus \{i\}\ \}$. It is easy to see that the sets $|F_i(t)|$ and $|C(t)|$ are related and on a complete influence graph it holds that $|C(t)| = |C(t)_{-i}| + |F_i(t)|$. On one hand, a best reply for which $|F_i(t-1)| = |F_i(t)|$ does not affect $|C(t)|$ since $|C(t-1)_{-i}| = |C(t)_{-i}|$. On the other hand, a best reply for which $|F_i(t-1)| > |F_i(t)|$ decreases the size of set C

$$\begin{aligned} |C(t)| &= |C(t)_{-i}| + |F_i(t)| = |C(t-1)_{-i}| + |F_i(t)| \\ &< |C(t-1)_{-i}| + |F_i(t-1)| = |C(t-1)| \end{aligned}$$

Since best replies for which $|F_i(t-1)| > |F_i(t)|$ do not exist in a cycle (*Part A*), best replies for which $|F_i(t-1)| < |F_i(t)|$ cannot exist either, as otherwise the size of set C would increase indefinitely. Hence $|F_i(t-1)| = |F_i(t)|$, which proves the lemma. □

Valentino Pacifici studied computer engineering at Politecnico di Milano in Milan, Italy. In October 2010, he completed a joint M.Sc. degree in computer engineering, between Politecnico di Milano and KTH, Royal Institute of Technology in Stockholm, Sweden. Now he is a researcher at the Laboratory for Communication Networks in KTH, Royal Institute of Technology and he is pursuing his Ph.D. His research interests include the modeling, design and analysis of content management systems using game theoretical tools.

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 and cyber-physical system security.

Paper **D**

Distributed Algorithms for Content Allocation in Interconnected Content Distribution Networks

Valentino Pacifici and György Dán

in Proc. of IEEE International Conference on Computer Communications (INFOCOM), 2015.

Distributed Algorithms for Content Allocation in Interconnected Content Distribution Networks

Valentino Pacifici and György Dán
ACCESS Linnaeus Center, School of Electrical Engineering
KTH, Royal Institute of Technology, Stockholm, Sweden
E-mail: {pacifici,gyuri}@kth.se

Abstract

Internet service providers increasingly deploy internal CDNs with the objective of reducing the traffic on their transit links and to improve their customers' quality of experience. Once ISP managed CDNs (nCDNs) become commonplace, ISPs would likely provide common interfaces to interconnect their nCDNs for mutual benefit, as they do with peering today. In this paper we consider the problem of using distributed algorithms for computing a content allocation for nCDNs. We show that if every ISP aims to minimize its cost and bilateral payments are not allowed then it may be impossible to compute a content allocation. For the case of bilateral payments we propose two distributed algorithms, the aggregate value compensation (AC) and the object value compensation (OC) algorithms, which differ in terms of the level of parallelism they allow and in terms of the amount of information exchanged between nCDNs. We prove that the algorithms converge, and we propose a scheme to ensure ex-post individual rationality. Simulations performed on a real AS-level network topology and synthetic topologies show that the algorithms have geometric rate of convergence, and scale well with the graphs' density and the nCDN capacity.

1 Introduction

Commercial content distribution networks (CDNs) have for a decade dominated the market of digital media delivery. For content providers they offer relatively low delivery costs compared to investing in an own infrastructure, they provide dynamically scaling bandwidth to satisfy sudden surges of demand, and through multiple surrogate servers they provide better quality of experience (QoE) for customers than a system based on a single content delivery server [1, 2].

As the digital media delivery market matures, major over-the-top content providers, such as Netflix, Hulu, etc, try to maintain customer satisfaction through increasing QoE; 3D content has become commonplace, and super HD content has become available recently [3]. Increased QoE results in increased bitrates, which stresses network operators' networks, yet in the traditional CDN-based content distribution model network operators are not part of the revenue chain. At the same time, good QoE may also require control of the network resources between the CDN surrogate and the customers' premises and needs content to be placed closer to the customers.

Many network providers have started to deploy their own CDNs for the above reasons, and recent industry efforts aim to interconnect these network provider managed CDNs (nCDNs), potentially also with traditional commercial CDNs [2, 4]. For content providers, nCDN interconnection provides a transparent solution for bringing content closer to the customers than any single CDN would be able to provide. For network providers, nCDN interconnection can improve CDN availability and customer QoE.

As nCDNs often prefetch content based on predicted demands during periods of low demands (e.g., NetFlix Open Connect), successful nCDN interconnection requires that given predicted demands, the nCDNs be able to agree on a content allocation that serves all service providers' interests. In lack of a central authority the agreement has to be based on a distributed algorithm, the algorithm should not reveal confidential information, and the resulting allocation should be such that no nCDN fares worse due to interconnection, as otherwise nCDNs would have no incentive to interconnect.

In this paper we address the design of distributed algorithms for content allocation among interconnected CDNs. We propose a model of CDN interconnection assuming that CDNs aim to maximize the QoE of their customers, and we show that self-enforcing content allocations may not exist if payments are not allowed among nCDNs. We propose two distributed algorithms that use bilateral compensations to guarantee convergence to a content allocation and we propose an opt-out scheme, which combined with the two algorithms ensures that the resulting allocations are individually rational. Thus, participation according to the proposed algorithms is *ex-post individually rational* for all nCDNs. We use simulations on a measured Internet AS-level topology to evaluate the proposed algorithms, and we show that faster convergence can be achieved if nCDNs reveal more private information, such as content demands. To the best of our knowledge ours is the first work to consider the design of ex-post individually rational distributed algorithms for CDN interconnection.

The rest of the paper is organized as follows. Section 2 describes the system model. In Section 3 we show that a satisfactory content allocation may not exist without payments. In Section 4 we design two distributed algorithms and we prove their convergence. Section 5 evaluates the proposed algorithms in terms of convergence rate and achieved cost savings. In Section 6 we review the related work. Section 7 concludes the paper.

2 System Model

We consider a set of autonomous service providers N. Each service provider manages a CDN; we refer to the CDN managed by service provider $i \in N$ as network CDN (nCDN) i. The customers of service provider i generate requests for content items from the set $\mathcal{O}$ of all content items. We make the common assumption that content is divisible in same-sized chunks, thus every item $o \in \mathcal{O}$ has the same size [5, 6]. The customers of service provider i generate requests for content item o at an average rate of $w_i^o \in \mathbb{R}_+$. We denote the set of content items stored by nCDN i by the set $A_i \in \mathcal{A}_i = \{A \subset \mathcal{O} : |A| = K_i\}$, where $K_i \in \mathbb{N}_+$ is the maximum number of items that nCDN i can store. In what follows, we use the terms *nCDN* and *service provider* interchangeably.

We model the relationships between the nCDNs by an undirected graph $\mathcal{G}(N, E)$, called the *interconnection graph.* There is an edge between nCDN i and nCDN j iff they are connected, and we use $\mathcal{N}(i) = \{j|(i,j) \in E\}$. We denote by A_{-i} the content item allocations of every nCDN other than nCDN i and by $\mathcal{R}_i(A_{-i})$ the set of items that can be retrieved from the nCDNs connected to nCDN i

$$\mathcal{R}_i(A_{-i}) \triangleq \bigcup_{j \in \mathcal{N}(i)} A_j. \tag{1}$$

We consider that each service provider aims to improve the quality of experience of its customers through decreasing the average access latency to content items. We denote by α_i the unit cost (i.e., access latency) of service provider i for serving an item stored in nCDN i, i.e., locally. If an item is not stored locally at nCDN i, it can be retrieved from one of the nCDNs $\mathcal{N}(i) \subset N$ connected to nCDN i. We denote by β_i^j the unit cost for serving an item from a connected nCDN $j \in \mathcal{N}(i)$. Observe that β_i^j depends on the neighbor j. If item o is available neither locally nor at a connected nCDN, it needs to be retrieved from the origin content provider in the network. We denote by γ_i the unit cost of retrieving an item from the origin content provider. We make the reasonable assumption that it is faster to access an item stored in the local nCDN than to retrieve it from a connected nCDN, and it is faster to retrieve an item stored in a connected nCDN than retrieving it from the content provider, i.e., $\alpha_i < \beta_i^j < \gamma_i$. This assumption is not restrictive, as if $\beta_i^j \geq \gamma_i$, we can remove (i,j) from E.

2.1 Average Access Latency Cost

We express the cost in terms of average access latency incurred by service provider i in allocation A as $C_i(A) = \sum_{o \in \mathcal{O}} C_i^o(A_i, A_{-i})$, where $C_i^o(A_i, A_{-i})$ is the cost for accessing item $o \in \mathcal{O}$,

$$C_i^o(A_i, A_{-i}) = w_i^o \begin{cases} \alpha_i & \text{if } o \in A_i \\ \min\limits_{j \in \mathcal{N}(i)} \{\beta_i^j | o \in A_j\} & \text{if } o \in \mathcal{R}_i(A_{-i}) \backslash A_i \\ \gamma_i & \text{otherwise.} \end{cases} \tag{2}$$

Observe that (i) the content allocations of the nCDNs in $\mathcal{N}(i)$ influence the cost of nCDN i through the set A_{-i}, and (ii) if item o is stored at several connected nCDNs then nCDN i retrieves it from the one with lowest unit cost. The cost incurred by service provider i for serving item o can be rewritten as

$$\begin{aligned} C_i^o(A_i, A_{-i}) &= C_i^o(\varnothing, A_{-i}) - (C_i^o(\varnothing, A_{-i}) - C_i^o(A_i, A_{-i})) \\ &= C_i^o(\varnothing, A_{-i}) - CS_i^o(A_i, A_{-i}), \end{aligned}$$

where $CS_i^o(A_i, A_{-i})$ is the cost saving that service provider i achieves by allocating item o given the content allocation at the nCDNs connected to nCDN i. Since the cost $C_i^o(\varnothing, A_{-i})$ is independent of the allocation A_i of nCDN i, finding the minimum cost is equivalent to finding the maximum aggregated cost saving

$$\begin{aligned} \arg\min_{A_i} C_i(A_i, A_{-i}) &= \arg\min_{A_i} \sum_o C_i^o(A_i, A_{-i}) \\ &= \arg\max_{A_i} \sum_o CS_i^o(A_i, A_{-i}). \end{aligned}$$

If nCDN i allocates item o, i.e., $o \in A_i$, then the cost saving can be rewritten as

$$CS_i^o(\{o\}, A_{-i}) = \begin{cases} w_i^o\,[\gamma_i - \alpha_i] & \text{if } o \notin \mathcal{R}_i(A_{-i}) \\ w_i^o\,[\beta_i^o(A_{-i}) - \alpha_i] & \text{if } o \in \mathcal{R}_i(A_{-i}) \end{cases} \tag{3}$$

where $\beta_i^o(A_{-i})$ is the lowest unit cost at which nCDN i can retrieve item o from a connected nCDN

$$\beta_i^o(A_{-i}) \triangleq \min_{j \in \mathcal{N}(i)} \{\beta_i^j | o \in A_j\}. \tag{4}$$

If instead $o \notin A_i$, the cost saving $CS_i^o(\{o\}, A_{-i}) = 0$. It follows that finding the minimum cost for service provider $i \in N$ corresponds to solving a knapsack problem where the values of the items are their cost savings given the allocations A_{-i} of the other nCDNs, and where the total weight is K_i.

2.2 Problem Statement

We consider that nCDNs would compute a content allocation based on predicted demands periodically, e.g., during epochs of low demand such as early mornings, and would perform prefetching to implement the allocation. We assume that bilateral payments are feasible, if necessary, and a payment $p_{i,j}$ would appear as an additive term in the cost function of nCDNs i and j. Payments can be settled periodically similar to peering agreements.

Without cooperation, i.e., if the set of connected nCDNs $\mathcal{N}(i) = \varnothing$ for every service provider i, service provider i would optimize the content allocation in nCDN i in isolation, and would prefetch the K_i items with highest demands. We denote the resulting allocation, which is *optimal in isolation*, by A_i^I. The corresponding cost is $C_i^I(A_i^I) = \sum_{o \in A_i^I} w_i^o \alpha_i + \sum_{o \in \mathcal{O} \setminus A_i^I} w_i^o \gamma_i$.

Cooperation could allow service providers to decrease their average access latency cost compared to isolation. For an allocation A we define the cost saving gain as

$$r_i(A) = \frac{C_i^I(\varnothing) - C_i(A)}{C_i^I(\varnothing) - C_i^I(A_i^I)} \tag{5}$$

where $C_i^I(\varnothing) = \sum_{o \in \mathcal{O}} w_i^o \gamma_i$ is the cost incurred by service provider i with no nCDN. We call an allocation A *individually rational* if $r_i(A) \geq 1$. Observe that service provider i benefits from cooperating only if $r_i(A) > 1$.

Since there is no central authority, cooperation requires a *distributed algorithm* that (i) requires information exchange between connected service providers only, (ii) reveals little private information such as content demands, and (iii) in a *finite* number of steps leads to a content allocation A that is *ex-post individually rational* for all service providers i.

3 Failure of Local Greedy Optimization

Without payments, a content allocation among interconnected nCDNs would have to let every nCDN i allocate content items that minimize its cost C_i, given the allocations of its connected nCDNs $\mathcal{N}(i)$. Such an allocation is self-enforcing, as no nCDN could gain by deviating from it. Modeling the interaction of nCDNs as a strategic game $\Gamma =< N, (\mathcal{A}_i)_{i \in N}, (U_i)_{i \in N} >$, where the utility of player i is the sum of its cost savings $U_i(A_i, A_{-i}) = \sum_o CS_i^o(A_i, A_{-i})$, such a content allocation corresponds to a pure strategy Nash equilibrium A^* of Γ, i.e., a set of allocations $(A_i^*)_{i \in N}$ such that

$$A_i^* \in \arg\max_{A_i} U_i(A_i, A_{-i}^*). \tag{6}$$

It is easy to see that A^* is individually rational.

Given an initial allocation of content items, $(A_i)_{i \in N}$, a distributed algorithm that might be used to compute A^* and one that reveals little private information is the *Local-Greedy* algorithm shown in Fig. 1. According to the *Local-Greedy* algo-

At time step t:

- nCDN $i_t \in N$ chooses a content allocation $A_{i_t}(t)$ such that $CS_{i_t}(A_{i_t}(t), A_{-i_t}(t-1)) > CS_{i_t}(A(t-1))$.
- nCDN i_t communicates to $\forall j \in \mathcal{N}(i_t)$ the set of items it plans to evict $E_{i_t}(t)$ and insert $I_{i_t}(t)$.

Figure 1: Pseudocode of the *Local-Greedy* algorithm.

rithm, at time step t a single nCDN i_t can update its allocation from $A_{i_t}(t-1)$ to an allocation $A_{i_t}(t)$ that increases its cost saving given the allocations of the other nCDNs $A_{-i_t}(t-1)$. The *Local-Greedy* algorithm requires little signaling: upon time step t nCDN i_t has to send the set $E_{i_t}(t) \triangleq A_{i_t}(t-1) \setminus A_{i_t}(t)$ of evicted and the set $I_{i_t}(t) \triangleq A_{i_t}(t) \setminus A_{i_t}(t-1)$ of inserted items to its neighboring nCDNs. The *Local-Greedy* algorithm terminates when no nCDN i can increase its cost saving by updating its allocation. By definition (6), if the *Local-Greedy* algorithm terminates, then the content allocation reached by the nCDNs is a pure strategy Nash equilibrium A^* of Γ. Nonetheless, it is not clear whether (i) an equilibrium allocation always exists and whether (ii) the *Local-Greedy* algorithm would lead to an equilibrium even if it exists.

In what follows we show that there are instances of the content allocation problem for which an equilibrium allocation A^* that satisfies (6) does *not exist.*

Non-Existence of Equilibrium Content Allocations

The strategic game Γ can be interpreted as a resource allocation game where the resources are the items, $c_i^o \triangleq w_i^o\,[\gamma_i - \alpha_i] \in \mathbb{R}_+$ is the value of resource o for player i and $0 < \delta_i^j \triangleq \frac{\beta_i^j - \alpha_i}{\gamma_i - \alpha_i} < 1$ is the penalty due to sharing the resource with player j. The expression of the cost saving in (3) becomes

$$CS_i^o(\{o\}, A_{-i}) = \begin{cases} c_i^o & \text{if } o \notin \mathcal{R}_i(A_{-i}) \\ c_i^o \min\limits_{j \in \mathcal{N}(i)} \{\delta_i^j | o \in A_j\} & \text{if } o \in \mathcal{R}_i(A_{-i}). \end{cases} \tag{7}$$

Observe that the cost incurred by player i for retrieving item o depends on which neighboring players store item o, not only on whether any neighboring player stores it as in [7, 8, 9]. As a consequence, results on the existence of Nash Equilibria in player-specific graphical congestion games do not apply.
Consider the following example.

Example 1. *Consider nCDNs $N = \{1, \ldots, 5\}$ and the set $\mathcal{O} = \{a, b, c, d\}$ of content items. The nCDNs are interconnected according to the graph in Figure 3a. For nCDN 5*

$$c_5^d \delta_4^5 > c_5^o \quad \forall o \in \mathcal{O} \setminus \{d\}. \tag{8}$$

For nCDN 1 the demands and the costs satisfy

$$\delta_1^2 < \delta_1^4,\ c_1^a < c_1^b, \tag{9}$$

$$c_1^b \delta_1^2 < c_1^a < c_1^b \delta_1^4. \tag{10}$$

Inequalities (9-10) specify a lattice (a poset with least and greatest element) over the cost savings $CS_1^o(\{o\}, A_{-1})$, which is shown in Figure 2a; we call it the cost saving graph. *An arrow between two cost savings points towards the greater of the two.*

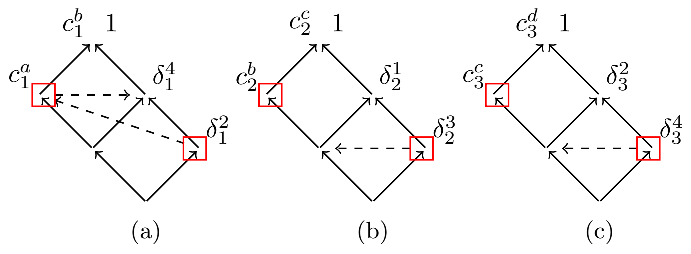


Figure 2: Cost saving graphs of nCDNs 1 to 3 in Example 1. The squares show the cost savings of each nCDN given the content allocation of its neighbors in the content allocation (a, b, c, d, d).

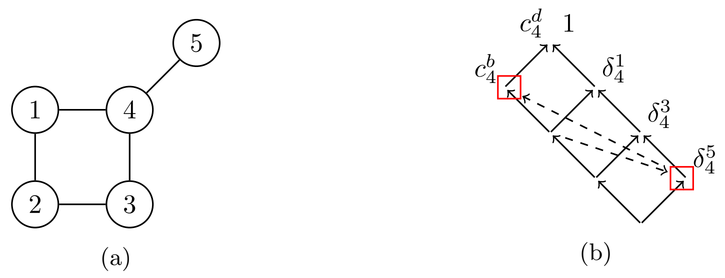


Figure 3: Interconnection graph (a) and cost saving graph (b) of nCDN 4 in Example 1. The squares show the cost savings of nCDN 4 in the content allocation (a, b, c, d, d).

The lattice is on the one hand the product of two totally ordered sets (solid arrows): values $\{c_i^o \ | o \in \mathcal{O}\}$ *and link costs* $\{\delta_i^j | j \in \mathcal{N}(i)\} \cup \{1\}$*. The greatest element of the lattice is the cost saving* c_i^o *of the item* $o \in \mathcal{O}$ *with highest rate* w_i^o *at nCDN* i *when it is not allocated by any connected nCDN* $j \in \mathcal{N}(i)$*, i.e.* $o \notin \mathcal{R}_i(A_{-i}) \Rightarrow CS_i^o(\{o\}, A_{-i}) = c_i^o$*. The least element of the lattice is the cost saving* $c_i^p \delta_i^j$ *of the item* $p \in \mathcal{O}$ *with lowest rate* w_i^p *when it is allocated by the connected nCDN* $j \in \mathcal{N}(i)$ *such that* $j = \arg\min_{k \in \mathcal{N}(i)} \delta_i^k$*. On the other hand, the lattice is specified through additional inequalities, such as (10) for nCDN* 1 *(dashed lines).*

The cost saving graphs for nCDNs $i \in \{2, 3, 4\}$ *are shown in Figure 2b, 2c and 3b, respectively. The squares in Figures 2 and 3b represent the cost savings* $CS_i^o(\{o\}, A_{-i})$ *of the corresponding nCDN* $i \in \{1, 2, 3, 4\}$ *at content allocation* $A = (a, b, c, d, d)$*. We omit the relations between cost savings that are not relevant for the example.*

We are now ready to prove the following.

Theorem 1. *An equilibrium content allocation that simultaneously minimizes the cost of every nCDN (pure Nash equilibrium) may not exist.*

Proof. We prove the theorem by showing that the game described in Example 1 does not possess a Nash equilibrium.
From (8) it follows that no content allocation A such that $A_5 \neq \{d\}$ is a Nash equilibrium. We can furthermore restrict the action set of each nCDN to the items that are included in its cost saving graph, according to the following

$$\mathcal{A}_1 = \big\{\{a\},\{b\}\big\} \quad \mathcal{A}_2 = \big\{\{b\},\{c\}\big\}$$
$$\mathcal{A}_3 = \big\{\{c\},\{d\}\big\} \quad \mathcal{A}_4 = \big\{\{d\},\{b\}\big\}$$

This results in a total of 16 possible content allocations. From the cost saving graphs in Figures 2 and 3b it follows that any content allocation where two interconnected nCDNs store the same item is not a Nash equilibrium; there are 12 such allocations. Therefore, it is enough to focus on the 4 content allocations in which there is no pair of interconnected nCDNs that store the same item.
The following sequence of content allocations starts with these four content allocations (marked with bold) and shows a cycling sequence of updates of nCDNs that follow the *Local-Greedy* algorithm. We omit nCDN 5, which always stores item (d).

$$\begin{array}{llllllll}
(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{d}) & \underset{4}{\rightarrow} & (\boldsymbol{a},\boldsymbol{b},\boldsymbol{c},\boldsymbol{b}) & \underset{3}{\rightarrow} & (\boldsymbol{a},\boldsymbol{b},\boldsymbol{d},\boldsymbol{b}) & \underset{2}{\rightarrow} & (\boldsymbol{a},\boldsymbol{c},\boldsymbol{d},\boldsymbol{b}) & \\
 & \underset{1}{\rightarrow} & (b,c,d,b) & \underset{4}{\rightarrow} & (b,c,d,d) & \underset{3}{\rightarrow} & (b,c,c,d) & \\
 & \underset{2}{\rightarrow} & (b,b,c,d) & \underset{1}{\rightarrow} & (a,b,c,d) & & & (11)
\end{array}$$

There is thus no NE, which proves the theorem. □

Observe that so far we have not assumed any relationship between the link costs δ_i^j and δ_j^i of interconnected nCDNs. There are recent measurement studies [10] that show that it may be reasonable to assume that link costs are symmetric between interconnected nCDNs, i.e., $\delta_i^j = \delta_j^i\ \forall j \in \mathcal{N}(i)$. In the case of Example 1, the requirement of symmetric link costs implies a feasible total order on the link costs, $1 > \delta_1^4 > \delta_1^2 > \delta_2^3 > \delta_3^4 > \delta_4^5$, which leads to the following.

Corollary 1. *An equilibrium content allocation that simultaneously minimizes the cost of every nCDN may not exist even if link costs are symmetric.*

A corresponding non-existence result for the case of a linear cost function and a directed interconnection graph was provided in [11]. Observe that, in our model, a link (i, j) is effectively directed if δ_i^j is sufficiently smaller than δ_j^i, such that the content allocated at nCDN i does not affect the allocation that minimizes the cost function of nCDN j. The importance of Corollary 1 is that it extends the non-existence results to undirected interconnection graphs.

Since an equilibrium allocation may not exist, Theorem 1 and Corollary 1 imply that if payments are not allowed then cost minimizing nCDNs may not be able to compute a content allocation, and algorithms like *Local-Greedy* may never terminate. Since it is infeasible to determine a-priori whether an equilibrium allocation exists (for reasons of computational complexity and because doing so would require global knowledge), computing a content allocation must involve payments to guarantee finite execution time.

4 Bilateral Compensation-based Allocation

A natural solution involving payments would be to model the problem as a coalition game with transferable utility, define the value function of a coalition as the maximum cost saving achievable by the set of players that form the coalition, and use the Shapley value for computing compensations. Since this value function is super-additive, the Shapley value is individually rational [12]. Nonetheless, such a solution is infeasible for several reasons. Computing the value of a coalition requires a single entity to know all item popularities. Second, it follows from the cost function (2) that computing the value of a coalition is NP-hard, and the computation needs to be done for all coalitions that induce a connected subgraph in $\mathcal{G}$.

In the following, we propose two distributed algorithms that involve *bilateral* compensations for computing an individually rational content allocation. The two algorithms differ in the amount of revealed private information, in the level of parallelism that they allow and, as we will see, in terms of convergence rate.

4.1 Aggregate-value Compensation Algorithm

Following the aggregate value compensation (AC) algorithm, at every time step t there is a set N_t of nCDNs that is allowed to update its content allocation. Given an allocation of content items $A(t-1)$, an update made by nCDN $i_t \in N_t$ from $A_{i_t}(t-1)$ to $A_{i_t}(t)$ can result in an increase of the cost (2) for one or more connected CDNs $j \in \mathcal{N}(i_t)$. According to the AC algorithm, an nCDN $j \in \mathcal{N}(i_t)$, $i_t \in N_t$, that would suffer an increase of cost $C_j(A_{N_t}(t), A_{-N_t}(t-1)) > C_j(A(t-1))$, offers a compensation to an nCDN $i_t \in \mathcal{N}(j) \cap N_t$ equal to its cost increase $\Delta C_j(t) \triangleq C_j(A_{i_t}(t), A_{-i_t}(t-1)) - C_j(A(t-1))$. We use $D_t \subseteq \mathcal{N}(i_t)$ to denote the set of nCDNs that offer a compensation,

$$j \in D_t \Leftrightarrow \Delta C_j(t) > 0. \qquad (12)$$

The compensations are used to deter nCDNs from performing updates: nCDN $i_t \in N_t$ performs the update despite the offered compensation if the aggregate compensation offered by all connected nCDNs is lower than the gain it achieves from updating the content allocation from $A_{i_t}(t-1)$ to $A_{i_t}(t)$. We call this the *Aggregate-value Compensation* (AC) algorithm, and we summarize it in Figure 4.

At time step t:

- Every nCDN $i_t \in N_t$ computes a content allocation $A^P_{i_t}(t)$ s.t. $CS_{i_t}(A^P_{i_t}(t), A_{-i_t}(t-1)) > CS_{i_t}(A(t-1))$.
- Every nCDN $i_t \in N_t$ communicates to $\forall j \in \mathcal{N}(i_t)$ the set of items it plans to evict $E_{i_t}(t)$ and insert $I_{i_t}(t)$.
- Every nCDN $j \in \mathcal{N}(N_t)$ s.t. $\Delta C_j(t) > 0$ offers a compensation $p^{i_t}_j(t) = \Delta C_j(t)$ to one nCDN $i_t \in \mathcal{N}(j) \cap N_t$
- **If** $\sum_{j \in D_t} p^{i_t}_j(t) \geq -\Delta C_{i_t}(t)$, then nCDN i_t accepts the compensation and it does not make the update, i.e., $A_{i_t}(t) = A_{i_t}(t-1)$.
  Otherwise nCDN i_t refuses the compensation and updates its allocation from $A_{i_t}(t-1)$ to $A_{i_t}(t) = A^P_{i_t}(t)$.
- Every nCDN $i_t \in N_t$ communicates to $\forall j \in \mathcal{N}(i_t)$ its decision, i.e. whether $A_{i_t}(t) = A_{i_t}(t–1)$ or $A_{i_t}(t) = A^P_{i_t}(t)$.

Figure 4: Aggregate-value Compensation (AC) Algorithm

Observe that the AC algorithm does not specify how N_t is chosen at each time step t, nor how an nCDN $j \in D_t$ chooses the recipient nCDN i_t of its compensation.

Before proving convergence for specific choices of N_t, we make the following definition.

Definition 1. A sequence $N_t \subseteq N$, $t = 1, \ldots$ of sets of nCDNs is a complete sequence, if for all t and each nCDN $i \in N$ there exists a time step $t' > t$ such that $i \in N_{t'}$.

Asynchronous operation: Let us first consider that only one nCDN $i_t \in N_t$ is allowed to update its allocation at each time slot t. Thus, the sets N_{t_1}, N_{t_2}.. are singletons and nCDN i_t is the only recipient of the compensation of each nCDN $j \in D_t$. The following result shows that the AC algorithm converges if used asynchronously.

Theorem 2. *Let N_t be a complete sequence of singleton sets and every nCDN use the* AC *algorithm. We refer to this as the* 1-AC *algorithm. The* 1-AC *algorithm converges to an allocation $\boldsymbol{A}$ of content items to interconnected nCDNs in a finite number of time steps.*

Proof. We prove the theorem by showing that the aggregate cost $C(A) \triangleq \sum_{i \in N} C_i(A_i, A_{-i})$ strictly decreases at every update made by any nCDN following the 1-AC algorithm.

Consider a nCDN i_t that updates its content allocation from $A_{i_t}(t-1)$ to $A_{i_t}(t)$ at time step t. It follows from (2) that $C_k(A(t)) = C_k(A(t-1))$ for any nCDN $k \notin$

$\mathcal{N}(i_t)$. We can calculate the aggregate cost function $C(A(t))$ after the update of nCDN i_t as

$$C(A(t)) = C(A(t-1)) + \Delta C_{i_t}(t) + \sum_{j \in D_t} \Delta C_j(t) + \sum_{j \in \mathcal{N}(i_t) \setminus D_t} \Delta C_j(t).$$

From (12) it follows that $\sum_{j \in \mathcal{N}(i_t) \setminus D_t} \Delta C_j(t) \leq 0$. Moreover, since nCDN i_t refused the compensation offered by the connected nCDNs in D_t, it follows that $\Delta C_{i_t}(t) + \sum_{j \in D_t} \Delta C_j(t) < 0$. Hence, at every update of the 1-AC algorithm $C(A(t)) < C(A(t-1))$. Since the set of all content allocations is finite and the sequence N_t is complete, this proves the theorem. □

A significant shortcoming of the 1-AC algorithm is that it requires global synchronization. Furthermore, if nCDN i_t is chosen uniformly at random at every time step t, the probability that nCDN i_t can decrease its cost C_{i_t} by updating its content allocation $A_{i_t}(t-1)$ at time step t decreases as the 1-AC approaches allocation $\boldsymbol{A}$. As a consequence, the convergence of the 1-AC algorithm may be slow.

Plesiochronous operation: In the following we show that convergence can be guaranteed even if the sets N_t are not singletons. Before we formulate our result, we recall the following definition from graph theory.

Definition 2. A *k-independent set* $\mathcal{I}^k$ of a graph $\mathcal{G} = (N, E)$ is a subset $\mathcal{I}^k \subseteq N$ of the vertexes of $\mathcal{G}$ such that the distance between any two vertexes of $\mathcal{I}^k$ in $\mathcal{G}$ is at least $k+1$. We denote by $\mathfrak{I}^k$ the set of all the k-independent sets of the interconnection graph $\mathcal{G}$.

We can now prove the following.

Theorem 3. *Let N_t be a complete sequence of 2-independent sets and every nCDN use the AC algorithm. We refer to this as the $\mathcal{I}^2$-AC algorithm. The $\mathcal{I}^2$-AC algorithm converges to an allocation $\boldsymbol{A}$ of content items to interconnected nCDNs in a finite number of time steps.*

Proof. Consider a nCDN $j \in \mathcal{N}(i_t)$, connected to $i_t \in \mathcal{I}_t^2$. From the definition of 2-independent set follows that i_t is the only nCDN in $\mathcal{N}(j)$ that is allowed to update its content allocation A_{i_t} at time step t. Hence, it is possible to compute the aggregate cost function $C(A(t))$ from $C(A(t-1))$ as follows

$$\begin{aligned} C(A(t)) = C(A(t-1)) \ + & \\ + \sum_{i_t \in U_t} \Big(\Delta C_{i_t}(t) + \sum_{j \in D_t} \Delta C_j(t) + \sum_{j \in \mathcal{N}(i_t) \setminus D_t} \Delta C_j(t) \Big), & \end{aligned}$$

where $U_t \subseteq \mathcal{I}_t^2$ is the set of nCDNs i_t such that $A_{i_t}(t) \neq A_{i_t}(t-1)$. From the same argument in the proof of Theorem 2 it follows that $C(A(t)) < C(A(t-1))$ at every update of the $\mathcal{I}^2$-AC algorithm. □

At time step t:

- Every nCDN $i_t \in N_t$ computes a content allocation $A^P_{i_t}(t)$ such that $CS_{i_t}(A^P_{i_t}(t), A_{-i_t}(t-1)) > CS_{i_t}(A(t-1))$.
- Every nCDN $i_t \in N_t$ communicates to $\forall j \in \mathcal{N}(i_t)$ the set of items it plans to evict $E_{i_t}(t)$ and insert $I_{i_t}(t)$.
- Every nCDN $j \in \cup_{i_t \in N_t} \mathcal{N}(i_t)$ calculates

  $$\Delta\widetilde{C}^o_j(t) = C^o_j(A^P_{N_t}(t), A_{-N_t}(t-1)) - C^o_j(A(t-1))$$

  for all $o \in \cup_{N_t} E_{i_t}$. If $\Delta\widetilde{C}^o_j(t) > 0$, nCDN j offers to a nCDN $k \in N_t$ such that $o \in E_k$ and $\beta^k_j = \beta^o_j(A_{-k}(t))$ a compensation $p^o_{j,k}(t) \triangleq \Delta\widetilde{C}^o_j(t)$.
- **If** $\sum_{j \in D_{i_t}} \sum_{o \in E_{i_t}} p^o_{j,i_t}(t) \geq -\Delta C_{i_t}(t)$, then nCDN i_t accepts the offer and it does not make the update, i.e., $A_{i_t}(t) = A_{i_t}(t-1)$.
  Otherwise nCDN i_t refuses the offer and updates its allocation from $A_{i_t}(t-1)$ to $A_{i_t}(t) = A^P_{i_t}(t)$.
- Every nCDN $i_t \in N_t$ communicates to $\forall j \in \mathcal{N}(i_t)$ its decision, i.e. whether $A_{i_t}(t) = A_{i_t}(t–1)$ or $A_{i_t}(t) = A^P_{i_t}(t)$.

Figure 5: Object-value Compensation (OC) Algorithm

Since the 2-independent sets of $\mathcal{G}$ are typically small, the number of nodes that can make updates simultaneously is small, and thus the convergence rate of the $\mathcal{I}^2$-AC algorithm may be only marginally faster than that of 1-AC. The number of simultaneous updates could be increased by using 1-independent sets, i.e., $\mathcal{I}^1$-AC, but the convergence of the $\mathcal{I}^1$-AC algorithm can not be guaranteed. We therefore propose an alternative to the AC algorithm in the following.

4.2 Object-value Compensation Algorithm

The object value compensation (OC) algorithm, shown in Figure 5, is similar to the AC algorithm, the difference is that nCDNs offer a compensation for each individual object that is to be evicted, instead of offering a compensation for the set of objects to be evicted. As we will see this difference allows for significantly faster convergence, but at the price of revealing more information about content item popularities.

For the OC algorithm we can prove the following.

Theorem 4. *Let N_t be a complete sequence of 1-independent sets and every nCDN* *use the* OC *algorithm. We refer to this as the* $\mathcal{I}^1$-OC *algorithm. The* $\mathcal{I}^1$-OC *algorithm converges to an allocation* $\boldsymbol{A}$ *of content items to interconnected nCDNs in a finite number of time steps.*

Proof. Consider the compensation $p^o_{j,k}(t)$ offered by nCDN j to nCDN k for the eviction of item $o \in E_k$ at time step t. Substituting (3) in the expression of $\Delta\widetilde{C}^o_j(t)$ we obtain

$$p^o_{j,k}(t) = w^o_j \left[\beta^o_j \left((A^P_{\mathcal{I}^1_t}(t), A_{-\mathcal{I}^1_t}(t-1))\right) - \beta^o_j\left(A(t-1)\right)\right].$$

We call U_t the set of nCDNs that update their content allocation at time step t of the algorithm, i.e. $U_t = \{i_t \in \mathcal{I}^1_t | A_{i_t}(t) \neq A_{i_t}(t-1)\}$. Since $U_t \subseteq \mathcal{I}^1_t$, it follows from (4) that $\beta^o_j \left((A^P_{\mathcal{I}^1_t}(t), A_{-\mathcal{I}^1_t}(t-1))\right) \geq \beta^o_j\left((A^P_{U_t}(t), A_{-U_t}(t-1))\right)$ and thus

$$p^o_{j,k}(t) \geq C^o_j(A(t)) - C^o_j(A(t-1)). \tag{13}$$

In the following we use (13) to prove that $C(A(t)) < C(A(t-1))$ at every update of the $\mathcal{I}^1$-OC algorithm. We can express the aggregate cost change $\Delta C(t) = C(A(t)) - C(A(t-1))$ as

$$\Delta C(t) = \sum_{i_t \in U_t} \Delta C_{i_t}(t) + \sum_{j \in D_t} \Delta C_j(t) + \sum_{j \in \mathcal{N}(i_t)\setminus D_t} \Delta C_j(t). \tag{14}$$

From (13) it follows that the second term

$$\sum_{j \in D_t} \Delta C_j(t) = \sum_{j \in D_t}\sum_{o \in \mathcal{O}} \Delta C^o_j(t) \leq \sum_{j \in D_t}\sum_{o \in \mathcal{O}} p^o_{j,k}(t). \tag{15}$$

Substituting (15) into (14) we obtain

$$\begin{aligned}\Delta C(t) &\leq \sum_{i_t \in U_t} \Delta C_{i_t}(t) + \sum_{j \in D_t}\sum_{o \in \mathcal{O}} p^o_{j,k}(t)\\ &= \sum_{i_t \in U_t} \Big(\Delta C_{i_t}(t) + \sum_{j \in D_{i_t}}\sum_{o \in \mathcal{O}} p^o_{j,i_t}(t)\Big).\end{aligned}$$

Since every nCDN $i_t \in U_t$ refused the offer and updated its allocation, it holds that $\Delta C_{i_t}(t) + \sum_{j \in D_{i_t}} \sum_{o \in \mathcal{O}} p^o_{j,i_t}(t) < 0$ for all $i_t \in U_t$. Since the set of all content allocations is finite and the sequence N_t is complete, this proves the theorem. □

4.3 Achieving Individual Rationality

The proposed algorithms terminate in a finite number of time steps, but the resulting content allocation $\boldsymbol{A}$ may not be individually rational, i.e., for some nCDNs i

```
1. Set $\ell \leftarrow 1$ and $N_c^\ell \leftarrow N$
2. At round $\ell$:
     • The nCDNs in $N_c^\ell$ run algorithm AC or OC until it terminates, in allocation $\boldsymbol{A}$.
     • Set $A^\ell \leftarrow \boldsymbol{A}$ and $N_c^{\ell+1} \leftarrow \{i \in N_c^\ell | r_i(A^\ell) \geq 1\}$.
3. If $|N_c^\ell \setminus N_c^{\ell+1}| > 0$:
     • Set $\ell \leftarrow \ell + 1$ and go to step 2).
```

Figure 6: Opt out scheme

it may hold $r_i(\boldsymbol{A}) < 1$. The nCDNs $i \in \{i \in N | r_i(\boldsymbol{A}) < 1\}$ would not have an incentive to implement $\boldsymbol{A}_i$, and would instead implement A_i^I. The Opt out scheme, shown in Figure 6, allows these nCDNs to implement A_i^I instead of $\boldsymbol{A}_i$ and iteratively re-executes the distributed algorithm with the remaining nCDNs; hence the final allocation is individually rational.

Corollary 2. *The* Opt out *scheme reaches an individually rational content allocation $\bar{A}$ in a finite number of iterations.*

Proof. Observe that the Opt out scheme terminates in allocation $\bar{A} = (A^\ell_{N_c^\ell}, A^I_{N \setminus N_c^\ell})$ only if $N_c^\ell = N_c^{\ell+1}$ at step 3). This is true if either $|N_c^\ell| = 0$ or $r_i(A^\ell) < 1\ \forall i \in N_c^\ell$. In both cases $\bar{A}$ is individually rational. □

Thus the $\mathcal{I}^2$-AC and the $\mathcal{I}^1$-OC algorithms combined with the OPT-OUT scheme are ex-post individually rational distributed algorithms for computing content allocations in a finite number of time steps, without prior global knowledge of the item popularities.

5 Evaluation

We use simulations to validate the results in Section 4 and to evaluate the convergence rate and the achieved gains for the cooperating nCDNs.

We consider three network topologies for the evaluation. The first topology is based on the Internet's AS-level topology reported in the CAIDA dataset [13] as of 1 Nov. 2013. In order to have a fairly large interconnection graph, we consider the ASes in the CAIDA dataset that are in Europe. As very small ASs are unlikely to deploy their own CDNs, we only consider ASs that have more than 2^{16}

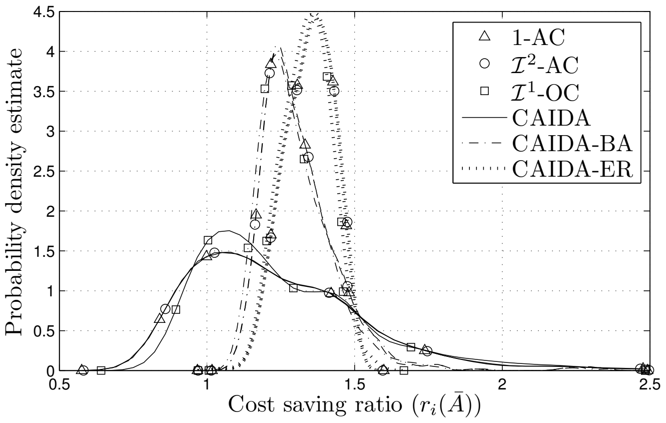


Figure 7: Probability density estimate of $r_i(\bar{A})$ for the three algorithms 1-AC, $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC on the CAIDA, CAIDA-BA and CAIDA-ER graphs. Results from 400 simulations.

IP addresses allocated. We consider two ASes connected if they have a business relationship (peering or transit) reported in the CAIDA dataset. We call *CAIDA graph* the largest connected component of the resulting topology, which consists of 638 ASes with an average node degree of 10.77. The other two topologies are Erdős-Rényi (CAIDA-ER) and Barabási-Albert (CAIDA-BA) random graphs that have same number of vertexes, average node degree and node degree ranking as the CAIDA graph. The node degree distributions of the three topologies do, however, differ in terms of their skeweness. We computed distance-1 and distance-2 colorings of all graph topologies by using the Welsh-Powell [14] and the Lloyd-Ramanathan [15] algorithms, respectively. We used $\alpha_i = 0.5$, $\gamma_i = 20$ at every nCDN and we computed the β_i^j as the propagation delay between nCDNs i and j assuming a signal propagation speed of $2 \cdot 10^5$ km/s. We considered $|\mathcal{O}| = 3000$ objects and the demands w_i^o for the content items at the various nCDNs follow Zipf's law with exponent 1. To simulate the algorithms, at each time step we choose an nCDN or a k-independent set uniformly at random, thus the sequence is complete. If not otherwise specified, the results shown are the averages of 200 simulations and $K_i = 20$ for every nCDN $i \in N$. We omit the confidence intervals as the results are within 5% of the average at a 0.95 confidence level.

5.1 Individual Rationality

We start with considering the gain of cooperation and the necessity of the Opt out scheme. Figure 7 shows the probability density estimate of the cost saving

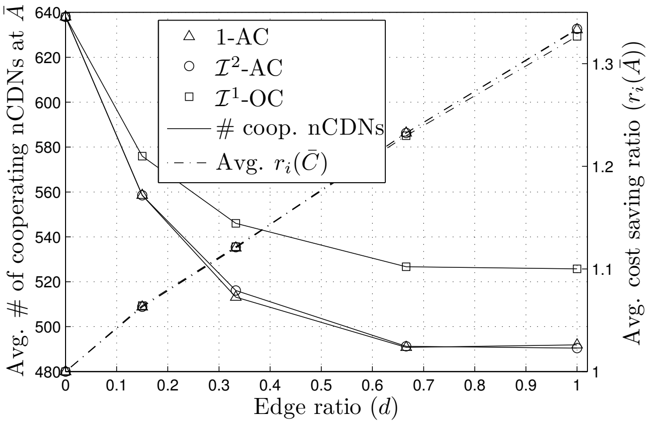


Figure 8: Average number of nCDNs choosing to cooperate and average cost saving ratio $r_i(\bar{A})$ at allocation $\bar{A}$, as a function of the edge ratio d for the three algorithms on the CAIDA graph.

ratios $r_i(A^{\ell=1})$ at the end of the first round of the Opt out scheme ($\ell = 1$) for all nCDNs for the three interconnection graphs and the three algorithms. The results show that the share of nCDNs for which the allocation is individually rational after the first round is determined by the graph topology. For the CAIDA-ER and the CAIDA-BA graphs, the content allocation is individually rational, $r_i(A^{\ell=1}) \geq 1$, for all nCDNs, and thus the Opt out scheme terminates after the first round, i.e. $\bar{A} = A^{\ell=1}$. On the contrary, for the CAIDA graph for many nCDNs $r_i(A^{\ell=1}) < 1$ after the first round. The difference is due to that the degree distribution of the CAIDA-BA graph is the most right-skewed among all the interconnection graphs, while the degree distribution of the CAIDA-ER graph is not skewed.

Observe that the probability densities for the 1-AC and $\mathcal{I}^2$-AC algorithms overlap, and are similar to that for the $\mathcal{I}^1$-OC algorithm. This suggests that the choice of the algorithm seems to have little impact on the gain from cooperation achieved by the nCDNs.

We evaluate the sensitivity of the results on synthetic topologies based on the CAIDA graph. The synthetic topologies were created by removing all edges from the CAIDA graph, and then reintroducing d fraction of the edges at random; the probability of reintroducing an edge between ASes i and j was proportional to the product of the number of IP addresses allocated to AS i and j.

Figure 8 shows the number of nCDNs choosing to cooperate and the average cost saving ratio $r_i(\bar{A})$ for the allocation $\bar{A}$ reached by the Opt out scheme, for the algorithms 1-AC, $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC on the CAIDA graph. The figure shows that the number of cooperating nCDNs is a decreasing convex function of the edge ratio d,

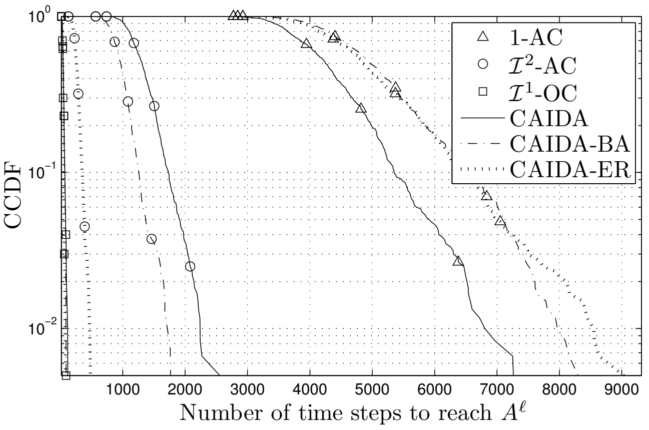


Figure 9: Complementary CDF of the number of time steps needed to reach allocation A^ℓ for the algorithms 1-AC, $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC on the CAIDA, CAIDA-BA and CAIDA-ER graphs.

suggesting that the majority of the nCDNs would not opt out from cooperation even if the graph was denser. Furthermore, the number of nCDN that would not opt out from cooperation is about 6% higher for the $\mathcal{I}^1$-OC algorithm compared to the $\mathcal{I}^2$-AC algorithm. At the same time the average cost saving ratio increases linearly, which is due to that the nCDNs have access to a linearly increasing amount of storage at neighbors.

5.2 Convergence Rate

We characterize the rate of convergence of the 1-AC, $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC algorithms by comparing the number of time steps needed to reach allocation A^ℓ during one round ℓ of the Opt out scheme. The number of time steps needed to reach A^ℓ is proportional to the time required by the algorithms to converge, as it also captures the parallelism embedded in the plesiochronous $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC algorithms.

Figure 9 shows the complementary CDF of the number of time steps needed to reach allocation A^ℓ based on 400 simulations for the three algorithms on the CAIDA, CAIDA-BA and CAIDA-ER graphs. The tail of each distribution decreases exponentially or faster as the number of time steps increases, which suggests that the rate of convergence is geometric. As expected, the 1-AC algorithm performs worst in terms of convergence rate, as it does not allow the nCDNs to update their allocations simultaneously. $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC are up to two orders of magnitude faster than 1-AC. Note that the fast convergence of the $\mathcal{I}^1$-OC algorithm is achieved at the price of increased information exchange between connected nCDNs compared to

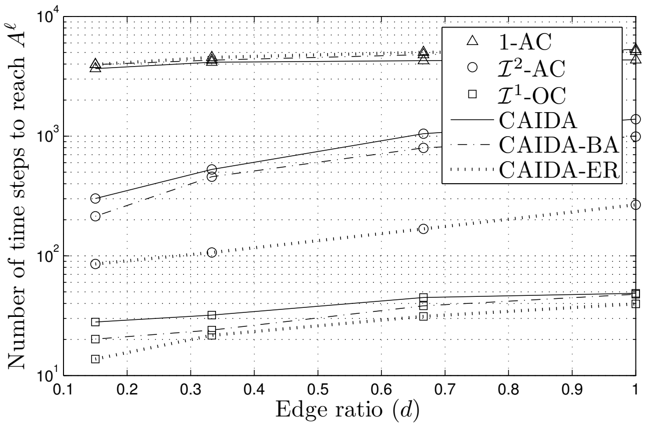


Figure 10: Average number of time steps needed to reach allocation A^ℓ as a function of the edge ratio d for the three algorithms on the CAIDA, CAIDA-BA and CAIDA-ER graphs.

the 1-AC and $\mathcal{I}^2$-AC algorithms. In practice, the object-wise information exchange between ASs may be problematic due to privacy concerns.

Figure 10 shows that the average number of time steps needed to reach allocation A^ℓ is an increasing concave function of the edge ratio d for all algorithms and interconnection graphs. Observe that the three interconnection graphs rank analogously for algorithms $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC but not for algorithm 1-AC. The reason lies in the average sizes of the k-independent sets used by the algorithms, which are reported in Table 1. The higher the average size of the k-independent sets, the higher the parallelism achieved by the $\mathcal{I}^k$-COMP algorithm, and the faster the convergence. As the coloring of the interconnection graph does not affect the performance of the 1-AC algorithm, the rankings of the three curves in both Figures 9 and 10 reflect other characteristics of the different network topologies.

Graph	#1-ind. sets	avg. size	#2-ind. sets	avg. size
CAIDA	16	39.8	219	2.9
CAIDA-BA	10	63.8	131	4.9
CAIDA-ER	8	79.8	36	17.8

Table 1: Number of k-independent sets and corresponding average size for the CAIDA, CAIDA-BA and CAIDA-ER interconnection graphs.

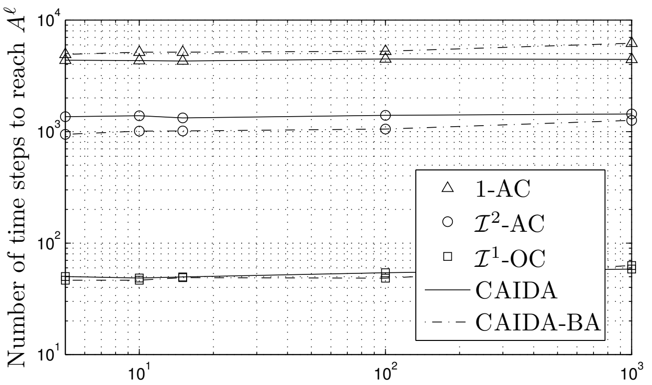


Figure 11: Average number of time steps needed to reach allocation A^ℓ as a function of the storage capacity K_i for the algorithms 1-AC, $\mathcal{I}^2$-AC and $\mathcal{I}^1$-OC on the CAIDA and CAIDA-BA graphs.

5.3 Scaling for Storage Capacity

In the following we investigate the effect of increasing the storage capacity K_i on the convergence rate of the proposed algorithms. Figure 11 shows the average number of time steps to reach allocation A^ℓ during one round ℓ of the OPT OUT scheme. We plot one curve for each algorithm on each of the CAIDA and CAIDA-BA graphs, as a function of the storage capacity K_i. The convergence rate is surprisingly insensitive to the storage capacity and the algorithms rank analogously to Figure 10. To explain this insensitivity we plot the average number of content item updates performed by the nCDNs for the same algorithms and graphs in Figure 12. We make two observations. First, the number of content item updates is the same for 1-AC and $\mathcal{I}^2$-AC, as they are both based on aggregate value compensation. The nCDNs perform less updates using the $\mathcal{I}^1$-OC algorithm, as they exchange object-wise compensations. The nCDNs perform less updates using the $\mathcal{I}^1$-OC algorithm, as they exchange object-wise compensations. Second, an nCDN can do an arbitrary number of content item updates during one time step, thus although the number of content items increases for larger storage sizes, this does not result in slower convergence in Figure 11.

6 Related Work

Our work is related to recent works on content placement in the context of CDNs [1, 16, 17, 18]. The majority of these works assume a single CDN operator and opti-

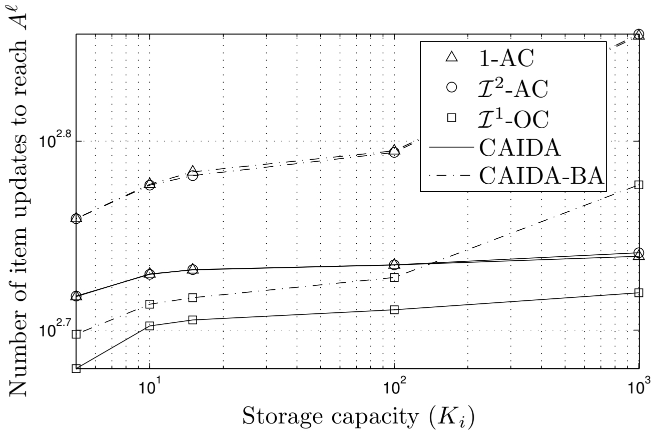


Figure 12: Average number of content item updates needed to reach allocation A^ℓ as a function of the storage capacity K_i for the three algorithms on the CAIDA and CAIDA-BA graphs.

mize content placement given a single performance objective. The authors in [16, 17] considered centralized algorithms for content placement and compared the retrieval cost for the different algorithms. Recently, [18] considered distributed algorithms that optimize for a single performance objective and provided analytical results for tree networks. In contrast to these works, in this paper we consider distributed algorithms for operator-managed CDNs, and thus the allocations need to be individually rational.

Closely related to ours are recent works on distributed selfish replication. Game theoretical analyses of equilibria and convergence for distributed selfish replication were considered in [7, 19, 20, 21, 22, 9]. The authors in [7] showed the existence of equilibria when the access costs are homogeneous and nodes form a complete graph. Similarly, [19, 20] assumed homogeneous costs and calculated the social cost of equilibria. The latter works considered that the nodes had no restriction on where to retrieve the content from. Other works [21, 22, 9] relax this assumption and introduce an interconnection graph to restrict the interaction between nodes. [21] assumed unit storage capacity and an infinite number of objects, showed the existence of equilibria and analyzed the price of anarchy for some special cases. [22] considered a variant of the problem where nodes can replicate a fraction of objects, and showed the existence of equilibria. The authors in [9] showed results in terms of convergence to equilibria in the case of homogeneous neighbor costs. In this paper we show that equilibrium existence results cannot be extended to the general problem of content replication on graphs and propose compensation-based algorithms that are guaranteed to converge.

Individually rational allocation of costs and revenues is the subject of cooperative game theory. Solution concepts such as the Shapley value and the core have found application in Internet routing [23] and in resource allocation [24], but these solution concepts require complete information and global enforcement, which make them impractical in the considered scenario. To the best of our knowledge this is the first work that proposes ex-post individually rational distributed algorithms for interconnected CDNs.

7 Conclusion

We considered the problem of computing a content allocation among interconnected autonomous CDNs. We showed that without payments there may be no allocation that minimizes the cost of all CDNs, and thus payments are necessary to guarantee that greedy algorithms would converge. For the case that payments are possible, we proposed two bilateral compensation-based distributed algorithms that are ex-post individually rational. The two algorithms require different amounts of information to be revealed by the CDNs, and allow different levels of parallelism. Numerical results show that the algorithms have geometric convergence, and that if CDNs reveal more private information about their content demands, the convergence of the algorithms becomes faster. Our results also show that the convergence times are fairly insensitive to the graph density and the amount of CDN storage.

References

[1] A.-M. K. Pathan and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks," The University of Melbourne, Tech. Rep., 2007.

[2] E. Bertrand, E. Stephan, T. Burbridge, P. Eardley, K. Ma, and G. Watson, "Use Cases for Content Delivery Network Interconnection," Internet Engineering Task Force (IETF), RFC, 2012. [Online]. Available: http://tools.ietf.org/html/rfc6770

[3] "Netflix." [Online]. Available: http://www.netflix.com

[4] L. Peterson and B. Davie, "Framework for CDN Interconnection," Internet Engineering Task Force (IETF), RFC, 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-cdni-framework/

[5] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate Models for General Cache Networks," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.

[6] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proc. of the 24th International Teletraffic Congress*, 2012.

[7] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed selfish replication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 12, pp. 1401–1413, 2006.

[8] V. Pacifici and G. Dán, "Content-peering Dynamics of Autonomous Caches in a Content-centric Network," in *Proc. of IEEE INFOCOM*, 2013, pp. 1079 – 1087.

[9] ——, "Convergence in Player-Specific Graphical Resource Allocation Games," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 11, pp. 2190–2199, 2012.

[10] Y. Schwartz, Y. Shavitt, and U. Weinsberg, "On the Diversity, Stability and Symmetry of End-to-End Internet Routes," in *Proc. of IEEE INFOCOM Workshop*, 2010, pp. 1–6.

[11] V. Bilò, A. Fanelli, M. Flammini, and L. Moscardelli, "Graphical Congestion Games." in *Proc. of WINE*, vol. 5385, 2008, pp. 70–81.

[12] L. S. Shapley, "A Value for n-Person Games," in *Contributions to the Theory of Games*, H. W. Kuhn and A. W. Tucker, Eds., 1953, pp. 307–317.

[13] "CAIDA. Automated Autonomous System (AS) ranking." [Online]. Available: http://as-rank.caida.org/data/

[14] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967.

[15] E. Lloyd and S. Ramanathan, "On the complexity of distance-2 coloring," in *Proc. of International Conference on Computing and Information (ICCI)*, 1992, pp. 71–74.

[16] M. Korupolu and M. Dahlin, "Coordinated placement and replacement for large-scale distributed caches," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 6, pp. 1317–1329, 2002.

[17] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz, "Efficient and Adaptive Web Replication Using Content Clustering," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 979–994, 2003.

[18] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," in *Proc. of IEEE INFOCOM*, 2010, pp. 1478–1486.

[19] G. Pollatos, O. Telelis, and V. Zissimopoulos, "On the social cost of distributed selfish content replication," in *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, 2008, pp. 195–206.

[20] E. Jaho, M. Karaliopoulos, and I. Stavrakakis, "Social similarity favors cooperation: the distributed content replication case," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 601–613, 2013.

[21] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz, "Selfish caching in distributed systems: a game-theoretic analysis," in *Proc. of ACM symposium on Principles of Distributed Computing (PODC)*, 2004, pp. 21–30.

[22] G. Dán, "Cache-to-cache: Could ISPs cooperate to decrease peer-to-peer content distribution costs?" *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1469–1482, 2011.

[23] R. T. B. Ma, J. C. S. Lui, V. Misra, and D. Rubenstein, "Internet Economics: The Use of Shapley Value for ISP Settlement," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 775–787, 2010.

[24] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, 2007.

Paper **E**

Coordinated Selfish Distributed Caching for Peering Content-centric Networks

Valentino Pacifici and György Dán

submitted to IEEE/ACM Transactions on Networking (ToN), first revision.

Coordinated Selfish Distributed Caching for Peering Content-centric Networks

Valentino Pacifici and György Dán [*]

Abstract

A future content-centric Internet would likely consist of autonomous systems (ASes) just like today's Internet. It would thus be a network of interacting cache networks, each of them optimized for local performance. To understand the influence of interactions between autonomous cache networks, in this paper we consider ASes that maintain peering agreements with each other for mutual benefit, and engage in content-level peering to leverage each others' cache contents. We propose a model of the interaction between the caches managed by peering ASes. We address whether stable and efficient content-level peering can be implemented without explicit coordination between the neighboring ASes. We show that content-level peering leads to stable cache configurations, both with and without coordination. However, peering ISPs that coordinate to avoid simultaneous updates converge to a stable configuration more efficiently. Furthermore, if the content popularity estimates are inaccurate, content-level peering is likely to lead to cost efficient cache allocations. We validate our analytical results using simulations on the measured peering topology of more than 600 ASes.

Index terms— Content-centric networks, cache networks, autonomous caches, content peering, stable content allocations.

1 Introduction

Recent proposals to re-design the Internet with the aim of facilitating content delivery share the common characteristic that caches are an integral part of the protocol stack [1, 2, 3]. In these content-centric networks users generate interest messages for content, which are forwarded until the content is found in a cache or the interest message reaches one of the content's custodians. The resulting network is

[*]Manuscript submitted for review March 3, 2015; revised October 21, 2015. Valentino Pacifici and György Dán are with the Laboratory for Communication Networks, School of Electrical Engineering, and the ACCESS Linnaeus Centre, KTH Royal Institute of Technology. Osquldas väg 10, 10044 Stockholm, Sweden. E-mail: {pacifici,gyuri}@kth.se. The work was partly funded by the Swedish Research Council through projects 2010:581 and 621-2014-635.

often modeled as a network of interacting caches. Several recent works aimed at optimizing the performance of a cache network through dimensioning cache sizes as a function of their location in the cache network [4], by routing interest messages to efficiently find contents [5] and by tuning the cache eviction policies used by the individual caches [6, 7].

Similar to the structure of today's Internet, a future content-centric network is likely to be a network of autonomous systems (AS). ASes are typically profit seeking entities and use an interior gateway protocol (IGP) for optimizing their internal routes. Nevertheless, they maintain client-provider and peering business relations with adjacent ASes [8], and they coordinate with each other using the Border Gateway Protocol (BGP), which allows them to exchange reachability information with their neighbors. The effect of BGP coordination on the stability and performance of global IP routing has been extensively investigated, e.g., the negative impact of damping route flaps [9, 10], the number of updates needed for BGP convergence [11], and general conditions for cycle-free IP routes [12].

ASes are likely to play a similar role in a future content-centric Internet as they do today, and thus, instead of a single cache network dimensioned and managed for optimal global performance, the content-centric Internet will be a network of cache networks, each cache network optimized for local performance. In lack of a central authority capable of enforcing a globally optimal allocation of contents to caches, it will be the interaction between cache networks that will determine the global cache allocation. To make such a network of cache networks efficient, we need to understand the potential consequences of interaction between the individual cache networks in terms of stability and in terms of the convergence of the cache contents, and the potential impact of coordination between the networks of caches.

In this work we consider a network of ASes that maintain peering agreements with each other for mutual benefit. The traffic exchanged between the peering AS is charged less than the traffic that each AS exchanges with its transit provider. The ASes maintain their own cache networks, and they engage in *content-level peering* in order to leverage each others' cache contents, which in principle should enable them to decrease their transit traffic costs. The interaction between the caches could, however, lead to unforeseen instability and oscillations, as in the case of BGP. Thus, a fundamental question that one needs to answer is whether stable and efficient content-level peering can be implemented without explicit coordination between the neighboring cache networks.

In this paper we address this question by proposing a model of the interaction and the coordination between the caches managed by peering ASes. We show that, with or without coordination, content-peering leads to stable cache configurations. Furthermore, we investigate how the convergence speed and the efficiency of the caching decisions are affected by coordination. Finally, we give insight into the structure of the most likely cache allocations in the case of inaccurate estimation of the arrival rate of user requests. We illustrate the analytical results using simulations on the measured peering topology of more than 600 ASes.

The rest of the paper is organized as follows. In Section 2 we describe the system model. In Section 3 we consider caching under perfect information, and in Section 4 we consider the case of imperfect information. In Section 5 we present numerical results, and in Section 6 we review related work. Section 7 concludes the paper.

2 System Model

We consider a set N of autonomous ISPs. Each ISP $i \in N$ is connected via peering links to some ISPs $j \in N$. We model the peering links among ISPs by an undirected graph $\mathcal{G} = (N, E)$, called the *peering graph.* We call $\mathcal{N}(i)$ the set of neighbors of ISP $i \in N$ in the peering graph, i.e. $\mathcal{N}(i) = \{j|(i,j) \in E\}$. Apart from the peering links, every ISP can have one or more transit links.

2.1 Content Items and Caches

We denote the set of content items by $\mathcal{O}$. We follow common practice and consider that every item $o \in \mathcal{O}$ has unit size [13, 14], which is a reasonable simplification if content is divisible into unit-sized chunks. Each item $o \in \mathcal{O}$ is permanently stored at one or more *content custodians* in the network. We denote by $\mathcal{H}_i$ the set of items kept by the custodians within ISP i. Since the custodians are autonomous entities, ISP i cannot influence the set $\mathcal{H}_i$. Similar to other modeling works, we adopt the Independent Reference Model (IRM) [15, 13, 14] for the arrival process of interest messages for the items in $\mathcal{O}$ generated by the local users of the ISPs. Under the IRM, the probability that the next interest message at ISP i is for item o is independent of earlier events. An alternative definition of the IRM is that the inter-arrival time of interest messages for item o at ISP i follows an exponential distribution with distribution function $F_i^o(x) = 1 - e^{-w_i^o x}$, where $w_i^o \in \mathbb{R}_+$ is the average arrival intensity of interest messages for item o at ISP i.

Each ISP $i \in N$ maintains a network of content caches within its network, and jointly engineers the eviction policies of the caches, the routing of interest messages and the routing of contents via the caches to optimize performance. The set of items cached by ISP i is described by the set $\mathcal{C}_i \in \mathfrak{C}_i = \{\mathcal{C} \subset \mathcal{O} : |\mathcal{C}| = K_i\}$, where $K_i \in \mathbb{N}_+$ is the maximum number of items that ISP i can cache. A *summary cache* in each ISP keeps track of the configuration of the local caches and of the content stored in local custodians, it thus embodies the information about what content is available within ISP i. We call $\mathcal{L}_i = \mathcal{C}_i \cup \mathcal{H}_i$ the set of items available within ISP i.

We denote by $\alpha_i > 0$ the unit cost of retrieving an item from a local cache, and by β_i the unit cost of retrieving an item from a peering ISP $j \in \mathcal{N}(i)$. The traffic on the transit link is charged by volume with unit cost γ_i, and we make the reasonable assumption that $\gamma_i > \beta_i \geq \alpha_i$. A particular case of interest is when retrieving an item from a peering cache is not more costly than retrieving it locally, i.e., $\beta_i = \alpha_i$ for all $i \in N$. We refer to this case as *free peering.* The model of *free peering* is motivated by that, once a peering link has been established between two ISPs, there is no additional cost for traffic.

2.2 Content-peering

We consider that time is divided into time slots, and peering ISPs *synchronously exchange information* about the contents of their summary caches *periodically*, at the end of every time slot. Upon receiving an interest message for an item, ISP i consults its summary cache to see if the item is available locally. If it is, ISP i retrieves the item from its local cache. Otherwise, before ISP i would forward the interest message to its transit provider, it tries to leverage its neighbors' caches by consulting its most recent copy of the summary caches of its peering ISPs $\mathcal{N}(i)$. In case a peering ISP $j \in \mathcal{N}(i)$ is caching the item, ISP i forwards the request to ISP j and fetches the content. If not, the interest message is sent to a transit ISP through a transit link.

Using the above notation, and denoting by $\mathcal{C}_{-i}$ the set of the cache configurations of every ISP other than ISP i, we can express the cost of ISP i to obtain item $o \in \mathcal{O}$ as

$$C_i^o(\mathcal{C}_i, \mathcal{C}_{-i}) = w_i^o \begin{cases} \alpha_i & \text{if } o \in \mathcal{L}_i \\ \beta_i & \text{if } o \in \mathcal{R}_i \setminus \mathcal{L}_i \\ \gamma_i & \text{if } o \notin \mathcal{L}_i \cup \mathcal{R}_i, \end{cases} \tag{1}$$

where $\mathcal{R}_i = \bigcup_{j \in \mathcal{N}(i)} \mathcal{L}_j$ is the set of items ISP i can obtain from its peering ISPs. The total cost can then be expressed as

$$C_i(\mathcal{C}_i, \mathcal{C}_{-i}) = \sum_{o \in \mathcal{O}} C_i^o(\mathcal{C}_i, \mathcal{C}_{-i}) \tag{2}$$

$$= \alpha_i \sum_{\mathcal{L}_i} w_i^o + \beta_i \sum_{\mathcal{R}_i \setminus \mathcal{L}_i} w_i^o + \gamma_i \sum_{\mathcal{O} \setminus \{\mathcal{L}_i \cup \mathcal{R}_i\}} w_i^o, \tag{3}$$

which is a function of the cache contents of the peering ISPs $\mathcal{N}(i)$.

2.3 Caching Policies and Cost Minimization

A content item o that is not available locally is obtained from a peering ISP or through a transit link, and is a candidate for caching in ISP i. The cache eviction policy of ISP i determines if item o should be cached, and if so, which item $p \in \mathcal{C}_i$ should be evicted to minimize the expected future cost. There is a plethora of cache eviction policies for this purpose, such as Least recently used (LRU), Least frequently used (LFU), LRFU (we refer to [16] for a survey of some recent algorithms).

We model the decision whether to evict item $p \in \mathcal{C}_i$ based on the comparison of the cost C_i incurred by ISP i when caching item o in place of item $p \in \mathcal{C}_i$.

ISP i caches item $o \notin \mathcal{C}_i$ in place of item $p \in \mathcal{C}_i$ if

$$C_i(\mathcal{C}_i \setminus \{p\} \cup \{o\}, \mathcal{C}_{-i}) < C_i(\mathcal{C}_i, \mathcal{C}_{-i}), \tag{4}$$

which is equivalent to

$$C_i^o(\varnothing, \mathcal{C}_{-i}) - C_i^o(\{o\}, \mathcal{C}_{-i}) > C_i^p(\varnothing, \mathcal{C}_{-i}) - C_i^p(\{p\}, \mathcal{C}_{-i}).$$

Let $\Psi_i^q(\mathcal{C}_{-i}) \triangleq C_i^q(\varnothing, \mathcal{C}_{-i}) - C_i^q(\{q\}, \mathcal{C}_{-i})$ be the cost saving of ISP i for caching item $q \in \mathcal{O}$. We consider that, in order to make a caching decision, ISP i calculates the cost savings $\Psi_i^o(\mathcal{C}_{-i})$ and $\Psi_i^p(\mathcal{C}_{-i})$ based on its estimates $\overline{w}_i^o$ and $\overline{w}_i^p$ of the arrival intensities w_i^o and w_i^p for the item o to be cached and for the item p in the cache, respectively. We consider two cases with respect to the accuracy of the arrival intensity estimates.

Perfect information: Under perfect information $\overline{w}_i^o = w_i^o \ \forall o \in \mathcal{O}$.

Imperfect information: Under imperfect information we consider that the probability of misestimation decreases exponentially with the difference in cost savings, that is, for $\Psi_i^o(\mathcal{C}_{-i}) > \Psi_i^p(\mathcal{C}_{-i})$ we have

$$P(\overline{\Psi}_i^o(\mathcal{C}_{-i}) < \overline{\Psi}_i^p(\mathcal{C}_{-i})) \propto \epsilon e^{-\frac{1}{\nu}(\Psi_i^o(\mathcal{C}_{-i}) - \Psi_i^p(\mathcal{C}_{-i}))} \tag{5}$$

As an example, consider an ISP i and a content allocation $\mathcal{C}$ such that $\mathcal{R}_i(\mathcal{C}) = \varnothing$, then (5) would have the form

$$\begin{aligned} P(\overline{\Psi}_i^o(\mathcal{C}_{-i}) < \overline{\Psi}_i^p(\mathcal{C}_{-i})) &\propto \epsilon e^{-\frac{1}{\nu}(w_i^o[\gamma_i - \alpha_i] - w_i^p[\gamma_i - \alpha_i])} \\ &= \epsilon e^{-\frac{1}{\nu}[\gamma_i - \alpha_i](w_i^o - w_i^p)}. \end{aligned}$$

This method of modeling the misestimation probability is reasonable for both the LRU and the LFU cache eviction policies. Under LRU the cache miss rate was shown to be an exponentially decreasing function of the item popularity [14]. Under a perfect LFU policy, if we denote the interval over which the request frequencies are calculated by τ, then the estimate $\overline{w}_i^p$ follows a Poisson distribution with parameter $w_i^p \tau$. The difference $k = \overline{w}_i^o \tau - \overline{w}_i^p \tau$ of two estimates thus follows the Skellam distribution [17] with density function

$$f(k, w_i^o \tau, w_i^p \tau) = e^{-\tau(w_i^o + w_i^p)} \left(\frac{w_i^o}{w_i^p} \right)^{k/2} I_{|k|}(2\tau \sqrt{w_i^o w_i^p}),$$

where $I_{|k|}(.)$ is the modified Bessel function of the first kind. The probability of misestimation is $\sum_{k=-\infty}^{-1} f(k, w_i^o \tau, w_i^p \tau)$, which decreases exponentially in $w_i^o - w_i^p$ for $\tau > 0$.

3 Content-peering under Perfect Information

We start the analysis by considering the case of perfect information, that is, when the cache eviction policies are not prone to misestimation.

The key question we ask is whether the profit-maximizing behavior of the individual ISPs would allow the emergence of an equilibrium allocation of items. If an equilibrium cannot be reached then content-peering could potentially lead to increased costs for the peering ISPs, as shown by the following simple example in which every ISP evicts and fetches the same items repeatedly over transit connections, thereby increasing their traffic costs compared to no content-peering.

Example 1. *Consider two ISPs and $\mathcal{O} = \{1, 2\}$. Let $K_1 = K_2 = 1$. Without content peering both ISPs cache their most popular item and forward interest messages to their transit provider for the least popular item. Their cost is thus $C_i = \alpha_i w_i^{h_i} + \gamma_i w_i^{l_i}$, where $w_i^{h_i} > w_i^{l_i}$. With content peering, if the initial allocation strategies are $\mathcal{C}_1 = \mathcal{C}_2 = \{1\}$ and the cost for retrieving an item from a peer is small enough, i.e. $\beta_i \gtrsim \alpha_i$, then the cache contents of the ISPs will evolve indefinitely as $(\{1\}, \{1\}) \to (\{2\}, \{2\}) \to (\{1\}, \{1\})$, etc. The average cost for the ISPs is thus $C_i' = \beta_i \left(\frac{w_i^{h_i} + w_i^{l_i}}{2} \right) + \gamma_i \left(\frac{w_i^{h_i} + w_i^{l_i}}{2} \right) > C_i$.*

This simple example illustrates that content peering could potentially lead to undesired oscillations of the cache contents of the ISPs, with the consequence of increased traffic costs. Ideally, for a stationary arrival of interest messages the cache contents should stabilize in an equilibrium state that satisfies the ISPs' interest of traffic cost minimization. Such an allocation corresponds to a pure strategy Nash equilibrium of the strategic game $\Gamma =< N, (\mathfrak{C}_i)_{i \in N}, (C_i)_{i \in N} >$, in which each ISP i aims to minimize its own cost C_i defined in (3).

Definition 1. A cache allocation $\mathcal{C}^* \in \times_{i \in N} \mathfrak{C}_i$ is an equilibrium allocation (pure strategy Nash equilibrium) if no single ISP can decrease its cost by deviating from it, that is

$$\forall i \in N, \forall \mathcal{C}_i \in \mathfrak{C}_i : C_i(\mathcal{C}_i^*, \mathcal{C}_{-i}^*) \leq C_i(\mathcal{C}_i, \mathcal{C}_{-i}^*) \tag{6}$$

The strategic game Γ was extensively studied in [18], the following result follows from [18, Theorem 1],

Theorem 1. *In the case of perfect information there is at least one equilibrium allocation.*

In the following we propose three distributed algorithms that avoid the inefficient updates shown in Example 1 and allow the system to reach an equilibrium allocation of items from which no ISP has an interest to deviate.

3.1 Asynchronous (Async) Algorithm

Example 1 suggests that if the ISPs coordinate so that they do not update their cache configurations simultaneously, then they would converge to an allocation from which neither of them would have an interest to deviate. In the case of Example 1, such allocations are $(\{1\},\{2\})$ or $(\{2\},\{1\})$. This intuition provides the rationale for the Async algorithm.

We start the description of the algorithm with the following definition.

Definition 2. A sequence $N_t \subseteq N$, $t = 1, ...$ of sets of ISPs is a complete sequence, if for all time slots t and each ISP $i \in N$ there exists a time slot $t' > t$ such that $i \in N_{t'}$.

In the Async algorithm N_t is a singleton, and at each time slot t, the algorithm allows the unique ISP $i_t \in N_t$ to update the set of its cached content $\mathcal{C}_{i_t}$. ISP i_t can decide to insert in its cache the items that are requested by one or more of its *local* users during time slot t but were not cached at the beginning of the time slot. At the same time, the rest of the ISPs $N \setminus \{i_t\}$ are not allowed to update the set of their cached contents. The pseudocode of the Async algorithm for every time slot $t \geq 1$ is then the following:

```
Async Algorithm
1: INPUT: arbitrary cache allocation $\mathcal{C}(0)$
2: $t \leftarrow 1$
3: **At** time slot $t$ **do**
4:     Allow ISP $i_t \in N_t$ to change its cached items
       from $\mathcal{C}_{i_t}(t-1)$ to $\mathcal{C}_{i_t}(t)$
5:     **for all** $j \in N \setminus \{i_t\}$ **do** $\mathcal{C}_j(t) = \mathcal{C}_j(t-1)$ **end**
6:     At the end of the time slot inform the ISPs $j \in \mathcal{N}(i_t)$
       about the new cache contents $\mathcal{C}_{i_t}(t)$
7:     $t \leftarrow t+1$
8: **end**
```

Figure 1: Pseudo-code of the Asynchronous (Async) Algorithm

What we are interested in is whether ISPs following the Async algorithm would reach an equilibrium allocation from which none of them would like to deviate. If the Async algorithm reaches such an allocation, then it terminates, and no other cache update will take place.

In the following we provide a sufficient condition for the Async algorithm to reach an equilibrium allocation. We call the condition *efficiency*, and the condition concerns the changes that each ISP $i \in N$ can make to its cache configuration.

Definition 3. Consider the updated cache configuration $\mathcal{C}_{i_t}(t)$ of ISP i_t immediately after time slot t. Define the evicted set as $E_{i_t}(t) = \mathcal{C}_{i_t}(t-1) \setminus \mathcal{C}_{i_t}(t)$ and the inserted set as $I_{i_t}(t) = \mathcal{C}_{i_t}(t) \setminus \mathcal{C}_{i_t}(t-1)$. $\mathcal{C}_{i_t}(t)$ is an *efficient update* if for any $o \in I_{i_t}(t)$ and any $p \in E_{i_t}(t)$

$$C^o_{i_t}(\mathcal{C}(t)) + C^p_{i_t}(\mathcal{C}(t)) < C^o_{i_t}(\mathcal{C}(t-1)) + C^p_{i_t}(\mathcal{C}(t-1)) \tag{7}$$

The requirement of efficiency is rather reasonable. Given that the ISPs are profit maximizing entities, it is natural to restrict the changes in the cache configuration to rational changes, i.e., changes that actually lead to lower cost. The concept of efficient update is similar to the concept of better reply in learning in strategic games.

Definition 4. The cache configuration $\mathcal{C}_{i_t}(t)$ is a *better reply* for ISP i_t in the strategic game Γ if $C_{i_t}(\mathcal{C}(t)) < C_{i_t}(\mathcal{C}(t-1))$.

It follows from Definition 3 and Definition 4 that any efficient update is a better reply in the strategic game Γ, although the converse is not true, i.e., some evictions and insertions in a better reply might not be efficient.

We model the evolution of the system state (the set of cache allocations) under the ASYNC algorithm by a Markov chain P^A; the transition probability $P(X_{t+1} = \mathcal{C}'|X_t = \mathcal{C})$ between states $X_t = \mathcal{C}$ and $X_{t+1} = \mathcal{C}' \neq \mathcal{C}$ at time slot t is non-zero if and only if there exists exactly one ISP $i \in N$ such that $\mathcal{C}'_i \neq \mathcal{C}_i$ and the update $\mathcal{C}_i$ to $\mathcal{C}'_i$ by ISP i is an efficient update. In that case, the probability $P(X_{t+1} = \mathcal{C}'|X_t = \mathcal{C})$ is directly proportional to the probability that ISP i receives the interest messages necessary to update its allocation from $\mathcal{C}_i$ to $\mathcal{C}'_i$ during time slot t. Given that the distribution F_i^o of the inter-arrival times is exponential with parameter w_i^o,

$$P(X_{t+1} = \mathcal{C}'|X_t = \mathcal{C}) \propto \prod_{o \in \mathcal{C}'_i \setminus \mathcal{C}_i} \left[1 - e^{-w_i^o \Delta_t}\right].$$

Observe that the Markov chain P^A is not irreducible, as every equilibrium state is an absorbing state.

Theorem 2. *If every ISP performs only efficient updates, the* ASYNC *algorithm terminates in an equilibrium allocation with probability 1.*

Proof. We prove the theorem by showing that P^A is an absorbing Markov chain whose absorbing states are the equilibrium allocations defined in (6), therefore the probability that the ASYNC algorithm reaches an equilibrium allocation is 1.

Call $\mathfrak{C}^*$ the set of absorbing states of P^A. Observe that $\mathfrak{C}^*$ corresponds to the set of pure strategy Nash Equilibria of the strategic game $\Gamma =< N, (\mathfrak{C}_i)_{i\in N}, (C_i)_{i\in N} >$ that was shown to be weakly acyclic under best replies in [18]. Weak acyclicity implies that from any state $\mathcal{C}^0 \notin \mathfrak{C}^*$, there exists a finite sequence of cache allocations $\mathcal{C}^0, \mathcal{C}^1, ...\mathcal{C}^{K-1}, \mathcal{C}^K$, such that *1)* $\mathcal{C}^K$ is an equilibrium allocation and *2)* for each $k \in \{1, 2, .., K\}$ there exists exactly one ISP $i^k \in N$ s.t. $\mathcal{C}^k_{i^k} \neq \mathcal{C}^{k-1}_{i^k}$ and $\mathcal{C}^k_{i^k} = \arg\min_{\mathcal{C}_{i^k} \in \mathfrak{C}_{i^k}} C_{i^k}(\mathcal{C}_{i^k}, \mathcal{C}^{k-1}_{-i^k})$. It is easy to see that the update from $\mathcal{C}^{k-1}_{i^k}$ to $\mathcal{C}^k_{i^k}$ is an efficient update of ISP i^k. In the following we prove that, for each $k \in \{1, 2, .., K\}$, state $\mathcal{C}^k$ is accessible from state $\mathcal{C}^{k-1}$ in P^A.

Assume w.l.o.g, that at time slot t process P^A is at state $X_t = \mathcal{C}^{k-1}$. Call t' the smallest time slot such that $t' > t$ and $i_{t'} = i^k$. For each time slot u such that $u \geq t$ and $u < t'$, consider the arrival of interest messages for item o generated by the local users of ISP i_u, with intensity $w^o_{i_u}$. As every update of the cache allocation of

ISP i_u is triggered by an interest message sent by a local user, and given that the distribution $F^o_{i_u}$ of the inter-arrival times is exponential with parameter $w^o_{i_u}$, there is a non-zero probability $e^{-w^o_{i_u}\Delta_u}$ that item o is not requested during time slot u of length Δ_u.
It follows that, starting from state $X_t = \mathcal{C}^{k-1}$, the probability that process P^A reaches $X_{t'-1} = \mathcal{C}^{k-1}$ is at least

$$P(X_{t'-1}= \mathcal{C}^{k-1}|X_t= \mathcal{C}^{k-1}) \geq \prod_{u=t}^{t'-1}\left[\prod_{o\notin\mathcal{C}_{i_u}} e^{-w^o_{i_u}\Delta_u}\right] > 0.$$

Hence, the probability $P(\mathcal{C}^k|\mathcal{C}^{k-1})$ to reach state $\mathcal{C}^k$ from $\mathcal{C}^{k-1}$ under P^A is at least

$$\prod_{u=t}^{t'-1}\left[\prod_{o\notin\mathcal{C}_{i_u}} e^{-w^o_{i_u}\Delta_u}\right]\cdot \prod_{o\in\mathcal{C}^k_{i^k}\setminus\mathcal{C}^{k-1}_{i^k}}\left[1-e^{-w^o_{i^k}\Delta_t}\right] > 0. \tag{8}$$

The second term of (8) is a lower bound on the probability that during time slot t', ISP i^k receives the interest messages necessary to update its cache allocation from $\mathcal{C}^{k-1}_{i^k}$ to $\mathcal{C}^k_{i^k}$. It follows that P^0 is an absorbing Markov chain, which proves the theorem. □

3.2 Cache-or-Wait (CoW) Algorithm

A significant shortcoming of the Async algorithm is that in slot t it disallows any ISP $j \in N \setminus \{i_t\}$ to perform an update. As a consequence, one ISP can perform an update on average every $|N|$ time slots. This restriction would provide little incentive for ISPs to adhere to the algorithm. In the following we investigate the effects of relaxing the requirement of strict coordination by allowing multiple ISPs to perform efficient updates during the same time slot.

Example 1 suggests that the oscillating behavior of the cache content is a consequence of the simultaneous cache updates by *neighboring* ISPs. The Cache-or-Wait (CoW) algorithm only allows *non-neighboring* ISPs to perform simultaneous updates. Before we describe the CoW algorithm, let us recall the notion of an independent set.

Definition 5. We call a set $\mathcal{I} \subseteq N$ an *independent set* of the peering graph $\mathcal{G}$ if it does not contain peering ISPs. Formally

$$\forall i, j \in \mathcal{I}, j \notin \mathcal{N}(i).$$

We denote by $\mathfrak{I}$ the set of all the independent sets of the peering graph $\mathcal{G}$. Consider a sequence of time slots t and a complete sequence of independent sets $\mathcal{I}_1, \mathcal{I}_2, \ldots \in \mathfrak{I}$ indexed by t. At each time slot t we allow every ISP $i \in \mathcal{I}_t$ to update the set of its cached content $\mathcal{C}_i$. At the same time, ISPs $j \notin \mathcal{I}_t$ are not allowed to update the set of their cached contents. The pseudocode of the CoW algorithm for every time slot $t \geq 1$ is then the following:

```
CoW Algorithm
1: INPUT: arbitrary cache allocation $\mathcal{C}(0)$
2: $t \leftarrow 1$
3: At time slot $t$ do
4:     Allow ISPs $i \in \mathcal{I}_t$ to change their cached items
       from $\mathcal{C}_i(t-1)$ to $\mathcal{C}_i(t)$
5:     for all $j \notin \mathcal{I}_t$ do $\mathcal{C}_j(t) = \mathcal{C}_j(t-1)$ end
6:     At the end of the time slot inform the ISPs $j \in \mathcal{N}(i)$
       about the new cache contents $\mathcal{C}_i(t)$
7:     $t \leftarrow t+1$
8: end
```

Figure 2: Pseudo-code of the Cache-or-Wait (CoW) Algorithm

Theorem 3. *If every ISP performs only efficient updates, then the* CoW *algorithm terminates in an equilibrium allocation with probability 1.*

Proof. The proof follows the same arguments as the proof of Theorem 2. We model the evolution of the system state under the CoW algorithm by a Markov chain P^W. P^W has the same state space as P^A but differs in terms of transition probabilities. In P^W, the transition probability between states $\mathcal{C}$ and $\mathcal{C}' \neq \mathcal{C}$ is non-zero if and only if there exists $\mathcal{I} \in \mathfrak{I}$ such that for every ISP i for which $\mathcal{C}'_i \neq \mathcal{C}_i$ we have $i \in \mathcal{I}$, and the update from $\mathcal{C}_i$ to $\mathcal{C}'_i$ is an efficient update. In the following we calculate a lower bound on the probability $P(\mathcal{C}^k|\mathcal{C}^{k-1})$ under P^W, for each $k \in \{1, 2, .., K\}$ in the finite sequence of cache allocations $\mathcal{C}^0, \mathcal{C}^1, ...\mathcal{C}^{K-1}$.

Assume that, at time slot t, process P^W is at state $X_t = \mathcal{C}^{k-1}$. Call i^k the ISP s.t. $\mathcal{C}^k_{i^k} \neq \mathcal{C}^{k-1}_{i^k}$ and t' the smallest time slot such that $t' > t$ and $i^k \in \mathcal{I}_{t'}$. The probability that process P^W will be at state $X_{t'-1} = \mathcal{C}^{k-1}$ at the beginning of time slot t' is lower bounded by the probability that no ISP $j \in \{\mathcal{I}_u | u \geq t, u \leq t'-1\}$ receives an interest message that triggers an update, that is

$$P(X_{t'-1}{=}\,\mathcal{C}^{k-1}|X_t{=}\,\mathcal{C}^{k-1}) \geq \prod_{u=t}^{t'-1} \prod_{j \in \mathcal{I}_u} \left[\prod_{o \notin \mathcal{C}^k_j} e^{-w^o_j \Delta_u} \right] > 0.$$

Similarly, we can calculate a lower bound on the probability of the transition from state $\mathcal{C}^{k-1}$ to state $\mathcal{C}^k$ during time slot t' as

$$P(X_{t'} = \mathcal{C}^k | X_{t'-1} = \mathcal{C}^{k-1}) \geq \prod_{j \in \mathcal{I}_{t'} \setminus \{i^k\}} \left[\prod_{o \notin \mathcal{C}_j} e^{-w_j^o \Delta_{t'}} \right] \cdot \prod_{o \in \mathcal{C}_{i^k}^k \setminus \mathcal{C}_{i^k}^{k-1}} \left[1 - e^{-w_{i^k}^o \Delta_{t'}} \right] > 0.$$

The first term of the product is a lower bound on the probability that no ISP $j \in \mathcal{I}_{t'}$ other than i^k receives an interest message that triggers an update at time slot t'. The second term was introduced in the proof of Theorem 2.
Finally, we can express a lower bound on the probability $P(\mathcal{C}^k|\mathcal{C}^{k-1})$ as

$$\begin{aligned} P(\mathcal{C}^k|\mathcal{C}^{k-1}) \geq \quad & P(X_{t'-1} = \mathcal{C}^{k-1} | X_t = \mathcal{C}^{k-1}) \cdot \\ & P(X_{t'} = \mathcal{C}^k | X_{t'-1} = \mathcal{C}^{k-1}) > 0. \end{aligned}$$

This proves the theorem. □

The following corollary is a consequence of Theorem 3

Corollary 1. *If every ISP performs efficient updates, then the number of* time slots *needed by the* CoW *algorithm to reach an equilibrium allocation is finite with probability 1, and thus the number of efficient updates is finite with probability 1.*

In the following we prove a stronger result on the number of *efficient updates* required to reach an equilibrium allocation in the *free peering* case. Recall that, in the *free peering* case, $\alpha_i = \beta_i$ for all $i \in N$.

Theorem 4. *In the* free peering *case, if every ISP performs efficient updates then the* CoW *algorithm terminates in an equilibrium allocation after a finite number of efficient updates.*

Proof. We will prove the theorem by showing that there exists a global function $\Psi : \times_i(\mathfrak{C}_i) \to \mathbb{R}$ that strictly increases at every efficient update made by any ISP i following the CoW algorithm. We define $\Psi(\mathcal{C}) = \sum_{i \in N} \Psi_i(\mathcal{C})$, where $\Psi_i(\mathcal{C})$ is the cost saving of ISP i for allocation $\mathcal{C} = (\mathcal{C}_i, \mathcal{C}_{-i})$,

$$\Psi_i(\mathcal{C}) = \sum_{o \in \mathcal{C}_i} \Psi_i^o(\mathcal{C}_{-i}) = \sum_{o \in \mathcal{C}_i} \left(C_i^o(\varnothing, \mathcal{C}_{-i}) - C_i^o(\{o\}, \mathcal{C}_{-i}) \right).$$

Without loss of generality, consider the efficient update $\mathcal{C}_i(t)$ made by ISP $i \in \mathcal{I}_t$ at time slot t. In the following we show that $\Psi_j(\mathcal{C}_i(t), \mathcal{C}_{-i}(t-1)) \geq \Psi_j(\mathcal{C}(t-1))$ for all $j \in N$. Observe that it follows directly from the definition of $\Psi_i(\mathcal{C})$ that for ISP i

$$\Psi_i(\mathcal{C}_i(t), \mathcal{C}_{-i}(t-1)) > \Psi_i(\mathcal{C}(t-1)).$$

A) Consider $k \notin \mathcal{N}(i)$. Observe that the cost of ISP k is not a function of $\mathcal{C}_i$:

- if $k \notin \mathcal{I}$, ISP k does not make any efficient update at time slot t, thus $\Psi_k(\mathcal{C}_i(t), \mathcal{C}_{-i}(t-1)) = \Psi_k(\mathcal{C}(t-1))$;
- if $k \in \mathcal{I}$, $k \neq i$, Ψ_k is not influenced by $\mathcal{C}_i$.

B) Consider $j \in \mathcal{N}(i)$. Consider $o \in I_i(t)$ and $p \in E_i(t)$. From the cost function defined in (1) it follows that $C_i^p(t+1) \geq C_i^p(t)$. Substituting it in the definition of efficient improvement step in (7), it follows that $C_i^o(t) > C_i^o(t+1) \Rightarrow o \notin \mathcal{R}_i(t) \Rightarrow o \notin \mathcal{C}_j(t)$, thus $\Psi_j(\mathcal{C}_i(t), \mathcal{C}_{-i}(t-1))$ is not affected by item o.
Consider now item p:

- If $p \notin \mathcal{C}_j(t)$, then $\Psi_j(\mathcal{C}_i(t), \mathcal{C}_{-i}(t-1))$ is not affected by item p.
- If $p \in \mathcal{C}_j(t)$, then $\Psi_j(\mathcal{C}_i(t), \mathcal{C}_{-i}(t-1)) \geq \Psi_j(\mathcal{C}(t-1))$ (the inequality is strict if $p \notin \{\mathcal{H}_j \cup \mathcal{R}_j(t+1)\}$).

Therefore, the function Ψ increases strictly upon every efficient update. Since $\times_i(\mathfrak{C}_i)$ is a finite set, Ψ cannot increase indefinitely and the CoW algorithm must terminate in an equilibrium allocation after a finite number of efficient updates. Note that, in game theoretical terminology, the function Ψ is a *generalized ordinal potential function* for the strategic game Γ. □

Thus, a network of ISPs, in which only non-peering ISPs perform efficient updates during the same time slot, eventually reaches an equilibrium allocation. Since the number of independent sets equals at least $\chi(\mathcal{G})$, the chromatic number of the ISP peering graph, an ISP can perform an update on average up to every $\chi(\mathcal{G})^{th}$ time slots. In the worst case, for complete peering graphs, for which $\chi(\mathcal{G}) = |N|$, the CoW algorithm would be equivalent to the Async algorithm, hence the ISPs would have little incentive to adhere to the CoW algorithm.

3.3 Cache-no-Wait (CnW) Algorithm

In the following we investigate what happens if we allow every ISP $i \in N$ in the system to update the set of its cached content $\mathcal{C}_i$ during every time slot. The pseudo-code of the CnW algorithm for time slots $t \geq 0$ is shown in Figure 3.

Using the same arguments as in the proofs of Theorems 2 and 3 we can prove the following.

Theorem 5. *If every ISP performs only efficient updates,* CnW *terminates in an equilibrium allocation with probability 1.*

Proof. Call P^N the Markov chain that models the evolution of the system state under the CnW algorithm. In P^N the transition probability between states $\mathcal{C}$ and $\mathcal{C}' \neq \mathcal{C}$ is non-zero if and only if, for every $i \in N$ such that $\mathcal{C}'_i \neq \mathcal{C}_i$, the update $\mathcal{C}_i$ to

CnW Algorithm

```
1: INPUT: arbitrary cache allocation $\mathcal{C}(0)$
2: $t \leftarrow 1$
3: **At** time slot $t$ **do**
4:     Every ISP $i \in N$ is allowed to change its cached items
       from $\mathcal{C}_i(t-1)$ to $\mathcal{C}_i(t)$
5:     At the end of the time slot ISP $i$ informs
       the ISPs $j \in \mathcal{N}(i)$ about the new cache contents $\mathcal{C}_i(t)$
6:     $t \leftarrow t+1$
7: **end**
```

Figure 3: Pseudo-code of the Cache-no-Wait (CnW) Algorithm

$\mathcal{C}_i'$ by ISP i is an efficient update. The probability $P(\mathcal{C}^k|\mathcal{C}^{k-1})$ under P^N, for each $k \in \{1, 2, .., K\}$ is at least

$$\prod_{o \in I_i^k} \left[1 - e^{-w_i^o \Delta_t}\right] \cdot \prod_{j \in N \setminus \{i\}} \prod_{o \notin \mathcal{C}_j} e^{-w_j^o \Delta_t} > 0,$$

where $I_i^k = \mathcal{C}_i^k \setminus \mathcal{C}_i^{k-1}$. It follows that P^0 is an absorbing Markov chain, which proves the theorem. $\square$

We have thus far shown that under perfect information content-level peering will eventually lead to stable cache allocations, independently of whether the ISPs coordinate. In the case of *free peering* we could provide a stronger result, as ISPs that coordinate are guaranteed to reach a stable allocation after a finite number of updates. We will later, in Section 5, investigate using simulations how the convergence speeds differ depending on the frequency of coordination. We now turn to the case of imperfect information.

4 The case of Imperfect Information

Until now we assumed that following a cache miss, when ISP i has to decide whether to cache item o it has a perfect estimate of the arrival intensity w_i^o of every item, and thus it is always able to evict one of the items that yield the lowest cost saving. In the following we consider that the estimation of the item popularities is imperfect. To ease the analysis we consider the Async algorithm throughout the section as the set of equilibria under the Async algorithm coincides with those under the CoW and CnW algorithms.

Under imperfect information the system can not settle in any single equilibrium or stable allocation, unlike in the case of perfect information. Nevertheless, the cache allocations that are most likely to occur are not arbitrary, and in the following we show that in some cases, it is possible to characterize them.

Call P^{ε} the Markov process that models the system under imperfect information, where $\varepsilon = e^{-\frac{1}{\nu}}$ is a scalar parameter that indicates the overall level of noise. Let us recall the following definition from [19].

Definition 6. A Markov process P^{ε} is a regular perturbation of the Markov process P^{A} if ε takes on all values in some interval $(0, a]$, and the following conditions hold for all $\mathcal{C}, \mathcal{C}' \in \times_{i \in N} \mathfrak{C}_i$:

1. P^{ε} is aperiodic and irreducible for all $\varepsilon \in (0, a]$,
2. $\lim_{\varepsilon \to 0} P^{\varepsilon}_{\mathcal{C},\mathcal{C}'} = P^{A}_{\mathcal{C},\mathcal{C}'}$,
3. $P^{\varepsilon}_{\mathcal{C},\mathcal{C}'} > 0$ for some ε implies $\exists r(\mathcal{C},\mathcal{C}') \geq 0$ such that $0 < \lim_{\varepsilon \to 0} \varepsilon^{-r(\mathcal{C},\mathcal{C}')} \cdot P^{\varepsilon}_{\mathcal{C},\mathcal{C}'} < \infty$.

We can now prove the following.

Theorem 6. *The Markov process P^{ε} is a regular perturbation of the Markov process P^{A}.*

Proof. 1) It follows from (5) that under imperfect information the probability that item o will be evicted and item p inserted even though $\Psi^{o}_{i} > \Psi^{p}_{i}$ is non-zero, therefore for every $\varepsilon > 0$, P^{ε} is irreducible. Furthermore, it follows from the distribution F^{o}_{i} of the inter-arrival times that $P(X_{t+1} = \mathcal{C} | X_t = \mathcal{C}) > 0$, $\forall \mathcal{C} \in \times_i(\mathfrak{C}_i)$, therefore P^{ε} is aperiodic.
2) From (5), for every $\mathcal{C}$ and $\mathcal{C}'$, P^{ε} converges to P^{A} at an exponential rate, $\lim_{\varepsilon \to 0} P^{\varepsilon}_{\mathcal{C},\mathcal{C}'} = P^{A}_{\mathcal{C},\mathcal{C}'}$.
3) For all $\mathcal{C}, \mathcal{C}' \in \times_{i \in N} \mathfrak{C}_i$, if $P^{\varepsilon}_{\mathcal{C},\mathcal{C}'} > 0$ then $\exists i \in N$ such that $\mathcal{C}_i \neq \mathcal{C}'_i$. We partition the inserted set $I = \mathcal{C}'_i \setminus \mathcal{C}_i$ in two sets, R and W, such that R is the set of content items that satisfy condition (7), and W is the set of objects mistakenly inserted. Call f an order of arrivals of the interest messages for the items in I, i.e. f is an ordering of the items in I. We define E_f as the set of items evicted upon the insertion of the items in W. It follows from (5) that the probability $P^{\varepsilon}(E_f \to W|f)$ of inserting the items in W in place of the items in E_f, given a particular order f of arrivals is

$$P^{\varepsilon}(E_f \to W|f) = \epsilon^{|W|} e^{-\frac{1}{\nu}\left(\sum_{o \in W} \Psi^{o}_{i}(\mathcal{C}_{-i}) - \sum_{p \in E_f} \Psi^{p}_{i}(\mathcal{C}_{-i})\right)}, \tag{9}$$

which does not depend on the order of eviction of the objects in E_f. We can then express $P^{\varepsilon}_{\mathcal{C},\mathcal{C}'}$ in the form

$$P^{\varepsilon}_{\mathcal{C},\mathcal{C}'} \propto \prod_{o \in I} \left[1 - e^{-w^{o}_{i}\Delta}\right] \sum_{f} P(f) P^{\varepsilon}(E_f \to W|f). \tag{10}$$

Observe that the probability $P(f)$ that the interest messages arrive in a particular order f is independent from the level of noise ε. Therefore, as $\varepsilon \to 0$, (10) is dominated by the term $P^\varepsilon(E_f \to W|f)$ with the largest exponent. By defining $r(\mathcal{C},\mathcal{C}') \triangleq \min_f \left(\sum_{o\in W} \Psi_i^o(\mathcal{C}_{-i}) - \sum_{p\in E_f} \Psi_i^p(\mathcal{C}_{-i})\right) > 0$, we have

$$0 < \lim_{\varepsilon\to 0} e^{\frac{1}{\nu} r(\mathcal{C},\mathcal{C}')} \cdot P^\varepsilon_{\mathcal{C},\mathcal{C}'} < \infty,$$

which proves the theorem. □

We refer to $r(\mathcal{C},\mathcal{C}') \geq 0$ as the resistance of the transition from allocation $\mathcal{C}$ to $\mathcal{C}'$. The resistance is 0 if there is a transition in the unperturbed Markov process (i.e., $W = \varnothing$). Since P^ε is an irreducible aperiodic finite Markov process, it has a unique stationary distribution for $\varepsilon > 0$. We now recall a result from Young [19].

Lemma 1 (Young [19])**.** *Let P^ε be a regular perturbed Markov process, and let μ^ε be the unique stationary distribution of P^ε for each $\varepsilon > 0$. Then $\lim_{\varepsilon\to 0}\mu^\varepsilon = \mu^A$ exists, and μ^A is a stationary distribution of P^A. The domain of μ^A is a non-empty subset of the absorbing states of the unperturbed Markov process.*

By Lemma 1 there is thus a stationary distribution μ^A of the unperturbed process such that, for small ε, the system will likely be in a state in the domain of μ^A. As the support of the stationary distribution μ^A is a subset of the recurrent communication classes of P^A, for small ε the perturbed process P^ε is likely to be in a particular subset of the equilibrium states. In the rest of the section we characterize the cache allocations that correspond to the equilibrium states that are most likely to be visited, for two scenarios.

4.1 The *free peering* case

We start by considering the *free peering* case, i.e. $\alpha_i = \beta_i$, for all $i \in N$. Observe that, at every ISP $i \in N$, the cost saving for every item $o \in \mathcal{R}_i(\mathcal{C})$ is $\Psi_i^o(\mathcal{C}_{-i}) = w_i^o[\beta_i - \alpha_i] = 0$. In other words, in the *free peering* case, no ISP i will ever insert any item o that is already cached by any of its peering ISPs $\mathcal{N}(i)$.
Consider a set of $N = \{1,\ldots,|N|\}$ ISPs, and items $\mathcal{O} = \{1,\ldots,|\mathcal{O}|\}$. Let $\rho_i(o)$ be the rank in terms of popularity of item o in ISP i, and let $\mathcal{T}_i$ be the set of the K_i items such that $\rho_i(o) \leq K_i$. For a cache allocation $\mathcal{C}$ denote by $h(\mathcal{C})$ the number of items o such that o is cached by an ISP i but $\rho_i(o) > K_i$.
We consider that the items with highest arrival intensity are the same among the different ISPs, and we denote them by the set $\mathcal{T} = \bigcup_i \mathcal{T}_i$. We start by investigating the cache allocations that are most likely to occur in the case of disjoint interests. In this case the K_i items with highest arrival intensity of the ISPs form disjoint sets, namely $\mathcal{T}_i \cap \mathcal{T}_j = \varnothing$, for all $i \neq j \in N$. We will first show the following

Lemma 2. *Let $\mathcal{C}^*$ be the allocation in which every ISP caches its most popular items, namely $\mathcal{C}_i^* = \mathcal{T}_i$. For any absorbing state $\mathcal{C}'$ such that $h(\mathcal{C}') > 2$, there exists an absorbing state $\mathcal{C}''$ such that $h(\mathcal{C}'') = 2$ and $r(\mathcal{C}^*,\mathcal{C}'') < r(\mathcal{C}^*,\mathcal{C}')$.*

Proof. Let $\mathcal{S}$ be the path with least resistance from $\mathcal{C}^*$ to $\mathcal{C}'$. Observe that, since $\mathcal{C}_i^* = \mathcal{T}_i$, at least $h(\mathcal{C}')$ mistakes are needed to reach $\mathcal{C}'$. Denote by i the first ISP that makes a mistake in $\mathcal{S}$, and by o and q the mistakenly evicted and inserted items, respectively. Since $\mathcal{S}$ is the path with least resistance, there exists $j \in \mathcal{N}(i)$ that makes at least one mistake. Consider the first mistake of ISP j and call p and r the evicted and inserted item, respectively. Observe that $o, p \in \mathcal{T}_i \cup \mathcal{T}_j$. We will now show that these two mistakes are enough to reach the absorbing state $\mathcal{C}''$ defined as $\mathcal{C}_j'' = \mathcal{C}_j^* \setminus \{p\} \cup \{o\}$, $\mathcal{C}_i'' = \mathcal{C}_i^* \setminus \{o\} \cup \{p\}$, $\mathcal{C}_h'' = \mathcal{C}_h^* \;\; \forall h \in N \setminus \{i, j\}$, and hence $r(\mathcal{C}^*, \mathcal{C}'') < r(\mathcal{C}^*, \mathcal{C}')$. Let us start from $\mathcal{C}^*$ and consider the state reached after committing the two mistakes. Observe that, since $q \notin \mathcal{T}_i \cup \mathcal{T}_j$, thus $\rho_i(p) < \rho_i(q)$. Furthermore we know that there is no ISP $h \neq j$, such that $p \in \mathcal{C}_h^*$. Hence ISP i can evict q and insert p without making a mistake. If $r = o$ then we reached $\mathcal{C}''$. If $r \neq o$ then, following the same argument, ISP j can insert o and evict r without making a mistake, reaching $\mathcal{C}''$. □

We will now use Lemma 2 to prove the following

Proposition 7. *If $\mathcal{T}_i \cap \mathcal{T}_j = \varnothing$ for all $i \neq j \in N$, then* $\lim_{\varepsilon \to 0} P(\mathcal{C}(t) = \mathcal{C}^*) = 1$.

Proof. As a consequence of Lemma 2 it is sufficient to show that for every absorbing state $\mathcal{C}''$ such that $h(\mathcal{C}'') = 2$, it holds that $r(\mathcal{C}^*, \mathcal{C}'') > r(\mathcal{C}'', \mathcal{C}^*)$. For brevity define $\mathcal{C}''$ as in the proof of Lemma 2. Assume, w.l.o.g., that in the path with least resistance from $\mathcal{C}^*$ to $\mathcal{C}''$, ISP i makes a mistake before ISP j by inserting item $q \notin \mathcal{T}_i \cup \mathcal{T}_j$ in place of item o . Then $r(\mathcal{C}^*, \mathcal{C}'') > w_i^o - w_i^q$. Observe now that, since $q \notin \mathcal{T}_i \cup \mathcal{T}_j$, from the absorbing state $\mathcal{C}''$ the mistake of ISP i of evicting item p and inserting q, with resistance $w_i^p - w_i^q$, is enough to reach $\mathcal{C}^*$. Hence $r(\mathcal{C}'', \mathcal{C}^*) \leq w_i^p - w_i^q$. This proves the Proposition. □

The following illustrates the proof on a simple example.

Example 2. *Consider a complete graph and $K_i = 1$. The $|N|$ most popular items are the same in every ISP, but item o has a distinct rank at every ISP. In every equilibrium the $|N|$ most popular items are cached, one at every ISP, and thus there are $|N|!$ equilibria. Fig. 4 shows the state transition diagram of the unperturbed Markov process (with solid lines) for the case of two ISPs, $|N| = 2$. The figure only shows the transitions* between *states. X and Y stand for an arbitrary item other than o and p, and the states (p, Y) and (X, p) ((o, Y) and (X, o)) represent all states in which item p (item o) is cached by ISP 1 and ISP 2, respectively. The dashed lines show transitions due to mistakes that are needed to move from one equilibrium to a state from which both equilibria are reachable (there is a positive probability of reaching it) in the unperturbed process. These transitions only exist in the perturbed Markov process. With perfect information there are two equilibrium allocations, which are the absorbing states (o, p) and (p, o) of the unperturbed process. The two equilibrium allocations are, however, not equally likely to be visited by the perturbed process.*

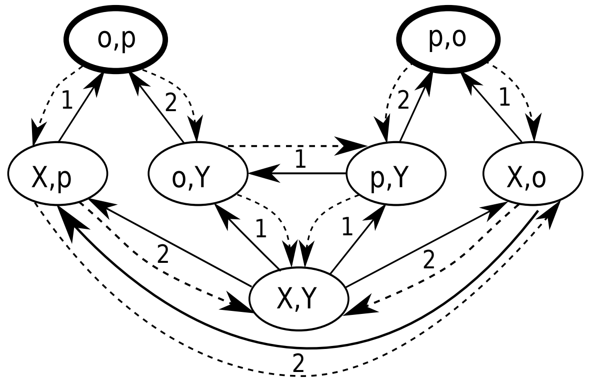


Figure 4: State transition diagram of the unperturbed Markov process (solid lines). (o, p) and (p, o) are absorbing states in the unperturbed Markov process, between the two equilibria. but only the equilibrium (o, p) is the domain of μ^A.

Observe that in the unperturbed process, equilibrium (o, p) is reachable from every allocation except from equilibrium (p, o). Therefore, in the perturbed process one mistake suffices to leave equilibrium (p, o) and to enter a transient state of the unperturbed process from which both equilibria are reachable in the unperturbed process. It takes, however, two mistakes in close succession to leave equilibrium (o, p) and to enter a transient state of the unperturbed process from which both equilibria are reachable in the unperturbed process. As the level of noise ε decreases, the probability of two successive mistakes decreases exponentially faster than that of a single mistake, and thus the perturbed process will be almost exclusively in state (o, p), thus $\mathcal{C}^* = (o, p)$.

A similar reasoning can be used to get insight into the evolution of the system state in the case that the ranking of the items is the same among all ISPs, namely $\mathcal{T}_i = \mathcal{T}_j$ for all $i, j \in N$. As an example, we show the following.

Proposition 8. *If the arrival intensity w_i^o for an item o for which $\rho_i(o) \leq K_i$ increases at ISP i, then $\lim_{\varepsilon \to 0} P(o \in \mathcal{C}_i(t))$ increases.*

Proof. Consider the state transition diagram of the perturbed Markov process P^ε. For a state $\mathcal{C}$ for which $o \in \mathcal{C}_i$, the transition probability that corresponds to ISP i mistakenly evicting o decreases. For a state $\mathcal{C}$ for which $o \notin \mathcal{C}_i$, the transition probability to the states $\mathcal{C}'$ for which $o \in \mathcal{C}_i'$ increases, and the transition probability to other states decreases. Reconciling these changes with the global balance equation for the set of states $\{\mathcal{C} | o \in \mathcal{C}_i\}$ proves the proposition. □

The impact of the number of peers of an ISP and that of the amount of storage K_i can be analyzed similarly. We omit the analysis for brevity, and turn to the general case instead.

4.2 The general case

In the following we consider the general case, when $\gamma_i > \beta_i \geq \alpha_i$. We show that the results of the previous section may not hold and show that Proposition 7 does not hold in general.
We start by considering a scenario similar to the one described in Example 2, when the K_i items with highest arrival intensity of the ISPs form disjoint sets.

Example 3. *Consider a complete peering graph of* 4 *ISPs and* $K_i = 1\ \forall i \in N$. *Let the most popular items of ISPs* 1, 2, 3, *and* 4 *be* a, o, p *and* b, *respectively. Thus, the allocation in which every ISP caches its most popular item is* $\mathcal{C}^* = (a, o, p, b)$. *Furthermore, let us assume that the following inequalities hold*

$$w_1^a [\beta_1 - \alpha_1] > w_2^q [\gamma_1 - \alpha_1],\ \forall q \in \mathcal{O} \setminus \{a\}, \tag{11}$$
$$w_2^p [\gamma_2 - \alpha_2] > w_2^r [\gamma_2 - \alpha_2] > w_2^o [\beta_2 - \alpha_2], \tag{12}$$
$$w_3^o [\gamma_3 - \alpha_3] > w_3^s [\gamma_3 - \alpha_3] > w_3^p [\beta_3 - \alpha_3], \tag{13}$$
$$w_4^b [\beta_4 - \alpha_4] > w_4^q [\gamma_4 - \alpha_4],\ \forall q \in \mathcal{O} \setminus \{b\}. \tag{14}$$

Observe that (11)-(14) imply that the allocation $\mathcal{C}'' = (a, p, o, b)$ *is an absorbing state, and* $h(\mathcal{C}'') = 2$. *Furthermore,* $\mathcal{C}^*$ *and* $\mathcal{C}''$ *are the only absorbing states.*

We are ready to prove the following

Proposition 9. *If* $\mathcal{T}_i \cap \mathcal{T}_j = \varnothing$ *for all* $i \neq j \in N$, *then there might exist two or more stochastically stable absorbing states, i.e.* $\lim_{\varepsilon \to 0} P(\mathcal{C}(t) = \mathcal{C}^*) < 1$.

Proof. We use Example 3 to prove the proposition. In particular, we show that for some values of w_1^o and w_4^p, the absorbing states $\mathcal{C}^*$ and $\mathcal{C}''$ have the same stochastic potential.
Consider the following two transitions: ISP 1 inserts item o in place of item a, and ISP 4 inserts item p in place of item b. We refer to the two transitions as (TI) and (T2), respectively.
We start by showing that there exists a path from $\mathcal{C}^*$ to $\mathcal{C}''$ in which the transitions (T1) and (T2) are the only transitions with positive resistance. Starting from $\mathcal{C}^*$, after transitions (T1) and (T2), the process reaches allocation (o, o, p, p). It follows from (11) and (14) that reaching $\mathcal{C}''$ does not require additional mistakes

$$\begin{aligned}(o, o, p, p) \underset{2}{\rightarrow} (o, r, p, p) \underset{3}{\rightarrow} (o, r, s, p) \underset{1}{\rightarrow} (a, r, s, p)\\ \underset{4}{\rightarrow} (a, r, s, b) \underset{2}{\rightarrow} (a, p, s, b) \underset{3}{\rightarrow} (\boldsymbol{a}, \boldsymbol{p}, \boldsymbol{o}, \boldsymbol{b}).\end{aligned}$$

Similarly, there exists a path from $\mathcal{C}''$ to $\mathcal{C}^*$ in which the transitions (T1) and (T2) are the only transitions with positive resistance. Starting from $\mathcal{C}''$, after transitions

(T1) and (T2), the process reaches allocation (o, p, o, p) and then reaches allocation $\mathcal{C}^*$ with no additional mistakes

$$\begin{aligned}(o,p,o,p) &\underset{2}{\rightarrow} (o,r,o,p) \underset{3}{\rightarrow} (o,r,s,p) \underset{1}{\rightarrow} (a,r,s,p) \\ &\underset{4}{\rightarrow} (a,r,s,b) \underset{2}{\rightarrow} (a,o,s,b) \underset{3}{\rightarrow} (\boldsymbol{a},\boldsymbol{o},\boldsymbol{p},\boldsymbol{b}).\end{aligned}$$

Since the peering graph is complete, $\mathcal{R}_1(\mathcal{C}^*) = \mathcal{R}_1(\mathcal{C}'') = \{\boldsymbol{o},\boldsymbol{p},b\}$ and $\mathcal{R}_4(\mathcal{C}^*) = \mathcal{R}_4(\mathcal{C}'') = \{\boldsymbol{o},\boldsymbol{p},a\}$. It follows that, starting from $\mathcal{C}^*$ or $\mathcal{C}''$, the resistances of transitions (T1) and (T2) are the same,

$$w_1^a\left[\gamma_1 - \alpha_1\right] - w_1^o\left[\beta_1 - \alpha_1\right] + w_4^b\left[\gamma_4 - \alpha_4\right] - w_4^p\left[\beta_4 - \alpha_4\right].$$

As the expression above does not depend on the average arrival intensities of interest messages at ISPs 2 and 3, there exist w_1^o and w_4^p such that the two paths described above are the paths with least resistance from $\mathcal{C}^*$ to $\mathcal{C}''$ and from $\mathcal{C}''$ to $\mathcal{C}^*$, respectively. It follows that the allocations $\mathcal{C}^*$ and $\mathcal{C}''$ have the same stochastic potential, that is $r(\mathcal{C}^*, \mathcal{C}'') = r(\mathcal{C}'', \mathcal{C}^*)$. This proves the proposition. □

It follows from Proposition 9 that if peering is not free, two or more cache allocations might be almost equally likely to emerge, even when the ISPs have disjoint interest.

5 Numerical Results

In the following we show simulation results to illustrate the analytical results of Sections 3 and 4 for CoW and CnW.

5.1 Perfect Information

Figures 5 and 6 show the average number of iterations and the average time the algorithms CoW and CnW need to terminate as a function of the time slot duration Δ, respectively. We report results for three different peering graphs. The CAIDA graph is based on the Internet AS-level peering topology in the CAIDA dataset [20]. The dataset contains 36878 ASes and 103485 transit and peering links between ASes as identified in [21]. The CAIDA graph is the largest connected component of peering ASes in the data set, and consists of 616 ISPs with measured average node degree of 9.66. The Erdős-Rényi (ER) and Barabási-Albert (BA) random graphs have the same number of vertexes and the same average node degree as the CAIDA graph. For the CoW algorithm, we used the Welsh-Powell algorithm to find a coloring [22] of the peering graph. We used $\alpha_i = 1$, $\beta_i = 1.5$, $\gamma_i = 10$ and cache capacity $K_i = 10$ at every ISP.
Each ISP receives interest messages for $|\mathcal{O}| = 3000$ items. The arrival intensities w_i^o follow Zipf's law with exponent 1, and for all $i \in N$ it holds $\sum_{o\in\mathcal{O}} w_i^o = 1$. Each

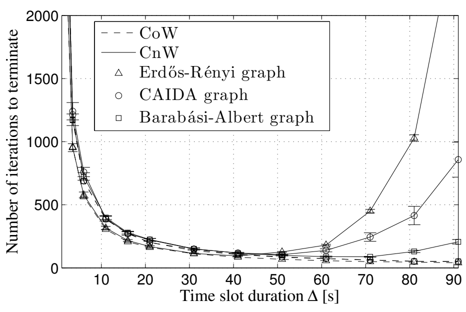


Figure 5: Average number of iterations needed to reach an equilibrium allocation as a function of the time slot duration Δ for three different peering graphs and algorithms CoW and CnW. Results for $\beta_i = 1.5$.

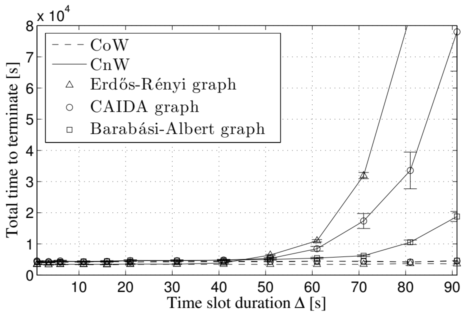


Figure 6: Average time needed to terminate as a function of the time slot duration Δ for three different peering graphs and algorithms CoW and CnW. Results for $\beta_i = 1.5$.

data point in the figures is the average of the results obtained from 100 simulations, and the error bars show the 95% confidence intervals. We omit the confidence intervals when they are within 5% of the averages.

Figure 5 shows that the number of iterations the CoW algorithm needs to reach an equilibrium allocation monotonically decreases with the time slot length. The longer the time slots, the more interest messages the ISPs receive within a time slot. This enables the ISPs to insert more highly popular objects per iteration. Furthermore, since only ISPs in an independent set can make updates at each iteration, simultaneous cache updates like the ones shown in Example 1 cannot occur. Consistently, the total time needed for the CoW algorithm to converge, shown in Figure 6, remains constant independent of the slot length Δ.

The CnW algorithm exhibits significantly different behavior for long time slots, as the number of iterations needed to terminate increases compared to the CoW algorithm. This happens because using the CnW algorithm a higher number of arrivals per time slot leads to a higher number of simultaneous updates, which disturb convergence. Figure 5 shows that simultaneous updates are most likely to occur in ER graphs. In BA graphs simultaneous updates would occur mainly among the few nodes with high degree, and since most ISPs have low node degree, the CnW algorithm would converge faster than on ER graphs. For the same reason, for small time slots when simultaneous updates are unlikely to occur, both the CoW and CnW algorithms perform best on the ER random graph. From Figure 6 we notice that, as expected, the time for the CnW algorithm to terminate starts to increase with high values of the slot length. This increase is fast for the ER graph due to the higher occurrence of simultaneous updates, as we discussed above.

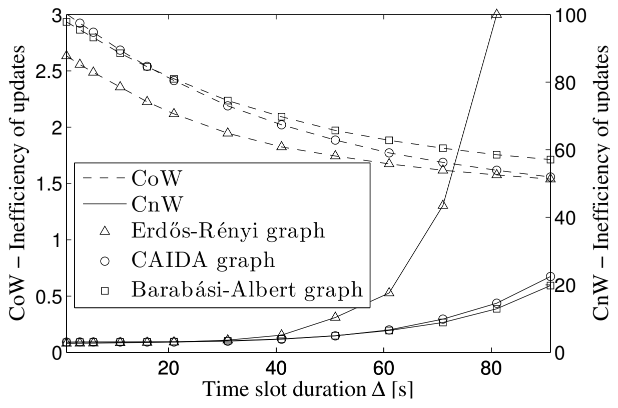


Figure 7: Average inefficiency as a function of the time slot duration Δ for three different peering graphs and algorithms CoW and CnW. Results for $\beta_i = 1.5$.

Figure 7 shows the number of items inserted in cache (potentially several times) for the two algorithms until termination divided by the minimum number of items needed to be inserted to reach the same equilibrium. We refer to this quantity as the *inefficiency of updates*. While the inefficiency of the CoW algorithm decreases slowly with the time slot length, that of the CnW algorithm shows a fast increase for high values of Δ, in particular for the ER and the BA graphs, which can be attributed to the simultaneous updates under CnW.

An important question is how the number of simultaneous updates that occur under CnW is affected by the peering cost β_i. Figure 8 shows the number of iterations needed to reach an equilibrium allocation under the CnW algorithm as a function of the time slot duration Δ. In the figure we show the results for peering costs $\beta_i \in \{1.0, 1.5, 2.0\}$. The results for $\beta_i = 1.5$ are the same as shown in Figure 5. The figure shows that the number of iterations decays as a power of the time slot duration, as long as convergence is not disturbed by simultaneous updates. At the same time, once simultaneous updates become frequent, increasing the time slot duration leads to a fast increase of the number of iterations. In the *free peering* case, i.e. $\beta_i = 1 (= \alpha_i)$, the number of iterations needed to reach an equilibrium allocation increases for significantly smaller time slot durations compared to the $\beta_i = 1.5$ and $\beta_i = 2.0$ cases. This happens because any item $o \in \mathcal{C}_i$ such that $o \in \mathcal{C}_j$ and $j \in \mathcal{N}(i)$ can potentially be evicted by both ISPs i and j, i.e. $\Psi_i^o(\mathcal{C}_{-i}) = \Psi_j^o(\mathcal{C}_{-j}) = 0$. This is not the case when $\alpha_i < \beta_i$, as higher peering costs correspond to higher cost savings for the items that are available at peering ISPs. Therefore, higher peering costs cause a lower number of simultaneous updates for equal time slot duration.

These results show that although CnW would be more appealing as it allows

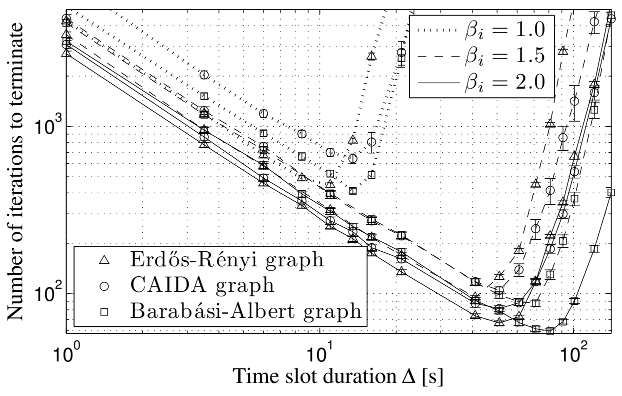


Figure 8: Average number of iterations needed to reach an equilibrium allocation as a function of the time slot duration Δ for algorithm CNW. Results for three different peering graphs and peering costs.

ISPs to update their cache contents all the time, CoW might be necessary when the information exchange happens infrequently or when the unit cost of peering traffic is close to the unit cost of retrieving an item from a local cache.

5.2 Imperfect Information

In the following we show results for the case when the estimation of the items' arrival intensities is imperfect under *free peering.* We consider that every ISP estimates the arrival intensities of the items by counting the number of arrivals under a period of τ seconds. As in the case of imperfect information the CoW algorithm would never terminate, we collected the statistics on the permanence of the various items in the cache of each ISP over 10^5 time slots. We considered 50 ISPs and a time slot of 70 seconds, which in the case of perfect information would guarantee a fast termination of the CoW algorithm. We first validate Proposition 7 for the case of $K_i = 1$, hence we consider that the item with the highest arrival intensity is different at every ISP.

Figure 9 shows the average relative permanence in the ISPs' caches of the three items with highest arrival intensity, as a function of the estimation interval τ, for three random peering graphs. The results show that the probability of caching the item with highest arrival intensity approaches 1 when τ increases, and thus validate Proposition 7. Furthermore we observe that the probability of caching items with lower arrival intensities decreases exponentially with τ.

In the next scenario we start from the setting described in Proposition 8, where the ranking of the items' intensities is the same among all ISPs. We scale the arrival intensity w_1^o of every item o at ISP 1 by the same factor, while keeping the intensities

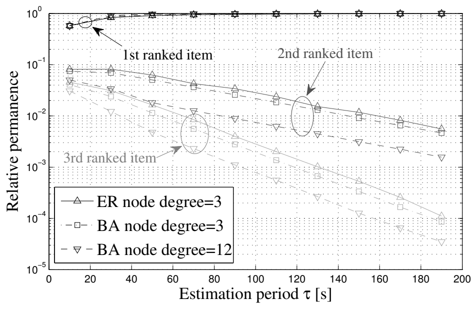


Figure 9: Relative permanence of the three items with highest arrival intensity in the ISPs' caches, as a function of the intensity estimation interval τ, for three different random peering graphs.

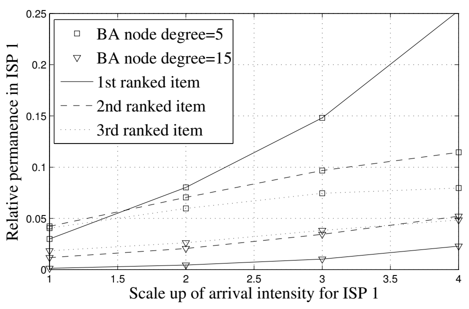


Figure 10: Relative permanence of the three items with highest arrival intensity in ISP 1's cache, as a function of the scaling factor at ISP 1, for two Barabási-Albert peering graphs with different average node degree.

at the other ISPs constant. Figure 10 shows the average relative permanence in ISP 1's cache of the three items with highest arrival intensities as a function of w_1. The results confirm that a higher w_1 leads to a higher relative permanence in the ISP's cache of the items with highest arrival intensity. Concerning the influence of the peering graph, the figure shows a constantly lower permanence of the best items for the BA graph with higher average node degree. This is due to that with a higher number of peering links the probability that the best items are in a peering ISP's cache gets higher.

6 Related Work

There is a large variety of cache eviction policies from Least recently used (LRU) to the recent Adaptive replacement cache [16]. Most analytical work on the performance of cache eviction policies for stand-alone caches focused on the LRU policy [23, 24, 14]. An iterative algorithm for calculating the cache hit rate was proposed in [23], closed-form asymptotic results were provided for particular popularity distributions in [24] and recently in [14]. These works considered stand-alone caches.

The cache hit rate for cache hierarchies was investigated in the context of web caches and content-centric networks [25, 26, 27, 28]. General topologies were considered for content-centric networks [4, 13, 6]. An iterative algorithm to approximate the cache miss rate in a network of caches was proposed in [13]. The authors in [4] considered various network topology-aware policies to improve the overall cache hit rate in a network of caches. In [6, 7] the authors use probabilistic caching to increase the cache hit rate and the fairness among content flows in a network of caches. The authors in [29] analyzed the impact of the initial system state on the selection of the system's steady-state. These works consider that the caches route requests irrespective of the associated traffic costs. [30] shows that considering the traffic costs of a network operator leads to cache allocations that are suboptimal in terms of hit rate. Common to the aforementioned works is the assumption of a single network operator with a single performance objective. In our work we account for the profit maximizing behavior of individual network operators and model the resulting interaction between caches.

Replication for content delivery in a hierarchy of caches was considered recently in [31, 32]. The authors in [31] considered a centralized algorithm for content placement, while distributed algorithms were analyzed in [32]. A game theoretical analysis of distributed content replication was provided in [33, 34, 18, 35]. The authors in [33] showed that the Price of Anarchy on a complete graph is unbounded and it depends on the unit costs to retrieve content items. [34] showed that, the social optimum may not be a Nash equilibrium even on a complete graph topology. In [18, 35] the authors investigated the existence of Nash equilibria and the convergence properties for the case of replication on an arbitrary graph topology, while [36] showed that a Nash equilibrium may not exist for certain graph topologies and access costs, and proposed a distributed algorithm for finding ex-post individually rational content allocations. Opposed to replication, we consider an arbitrary topology of caches, and we consider that caches do not follow an algorithm engineered for good global performance but they follow their individual interests.

Closest to this work is [37], which considered a network of selfish caches, including the effects of evictions, and provided a game-theoretical analysis of the resulting cache allocations. This paper extends the results in [37] by relaxing the assumption of free content-peering and considering the more general case when retreiving content from within the ISP is not more costly than retrieving it from peering ISPs.

7 Conclusion

We proposed a model of the interactions between the caches managed by peering ASes in a content-centric network. We used the model to investigate how the level of coordination influences the ability of peering ASs to achieve stable and efficient cache allocations in the case of content-level peering. We showed that irrespective of whether the ISPs coordinate, the cache allocations of the ISPs engaged in content-level peering will reach a stable state. If fast convergence to a stable allocation is important too then coordination is needed to avoid simultaneous cache evictions by peering ISPs. Furthermore, we gave insight into the structure of the most likely cache allocations for the case when the content popularity estimates are inaccurate. We showed that, in the general case, various cache allocations may be almost equally likely to emerge. However, if peering traffic is free, content-peering is likely to lead to the cache allocation that is most efficient.

References

[1] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A data-oriented (and beyond) network architecture,” in *Proc. of ACM SIGCOMM*, vol. 37, no. 4, 2007, pp. 181–192.

[2] C. Dannewitz, “NetInf: An Information-Centric Design for the Future Internet,” in *Proc. of GI/TG KuVS Work. on The Future Internet*, 2009.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *Proc. of ACM CoNEXT*, 2009.

[4] D. Rossi and G. Rossini, “On sizing CCN content stores by exploiting topological information,” in *Proc. of IEEE INFOCOM, NOMEN Workshop*, 2012, pp. 280–285.

[5] E. J. Rosensweig and J. Kurose, “Breadcrumbs: Efficient, Best-Effort Content Location in Cache Networks,” in *Proc. of IEEE INFOCOM*, 2009, pp. 2631–2635.

[6] I. Psaras, W. K. Chai, and G. Pavlou, “Probabilistic In-Network Caching for Information-Centric Networks,” in *ICN workshop*, 2012, pp. 1–6.

[7] ——, “In-Network Cache Management and Resource Allocation for Information-Centric Networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 11, pp. 2920 – 2931, 2014.

[8] P. Faratin, D. Clark, P. Gilmore, S. Bauer, A. Berger, and W. Lehr, “Complexity of Internet Interconnections : Technology , Incentives and Implications for Policy,” in *Proc. of Telecommunications Policy Research Conference*, 2007, pp. 1–31.

[9] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz, “Route Flap Damping Exacerbates Internet Routing Convergence,” in *Proc. of ACM SIGCOMM*, vol. 32, no. 4, 2002, p. 221.

[10] S. Agarwal, C.-N. Chuah, S. Bhattacharyya, and C. Diot, “The impact of BGP dynamics on intra-domain traffic,” in *Proc. of ACM SIGMETRICS*, vol. 32, no. 1, 2004, p. 319.

[11] R. Sami, M. Schapira, and A. Zohar, “Searching for Stability in Interdomain Routing,” in *Proc. of IEEE INFOCOM*, 2009, pp. 549–557.

[12] L. G. L. Gao and J. Rexford, “Stable Internet routing without global coordination,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 681–692, 2001.

[13] E. J. Rosensweig, J. Kurose, and D. Towsley, "Approximate Models for General Cache Networks," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.

[14] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proc. of the 24th International Teletraffic Congress (ITC)*, 2012, pp. 1–8.

[15] E. G. Coffman Jr. and P. J. Denning, *Operating Systems Theory*, 1973.

[16] N. Megiddo and D. Modha, "ARC: A Self-Tuning, Low Overhead Replacement Cache," in *Proc. of USENIX File & Storage Technologies Conference (FAST)*, 2003, pp. 115 – 130.

[17] J. G. Skellam, "The frequency distribution of the difference between two Poisson variates belonging to different populations." *Journal Of The Royal Statistical Society*, vol. 109, no. 3, p. 296, 1946.

[18] V. Pacifici and G. Dán, "Convergence in Player-Specific Graphical Resource Allocation Games," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 11, pp. 2190–2199, 2012.

[19] H. P. Young, "The evolution of conventions," *Econometrica: Journal of the Econometric Society*, vol. 61, no. 1, pp. 57–84, 1993.

[20] "CAIDA. Automated Autonomous System (AS) ranking." [Online]. Available: http://as-rank.caida.org/data/

[21] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. Claffy, and G. Riley, "AS relationships: inference and validation," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 29–40, 2007.

[22] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967.

[23] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proc. of ACM SIGMETRICS*, vol. 18, no. 1, 1990, pp. 143–152.

[24] P. R. Jelenković, "Asymptotic Approximation of the Move-to-Front Search Cost Distribution and Least-Recently Used Caching Fault Probabilities," *Annals of Applied Probability*, vol. 9, no. 2, pp. 430–464, 1999.

[25] M. Busari and C. Williamson, "Simulation Evaluation of a Heterogeneous Web Proxy Caching Hierarchy," in *Proc. of MASCOTS*, 2001, p. 379.

[26] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical Web caching systems," in *Proc. of IEEE INFOCOM*, vol. 3, 2001, pp. 1416–1424.

[27] C. Williamson, “On filter effects in web caching hierarchies,” *ACM Trans. Int. Tech.*, vol. 2, no. 1, pp. 47–77, 2002.

[28] I. Psaras, R. G. Clegg, R. Landa, W. K. Chai, and G. Pavlou, “Modelling and evaluation of CCN-caching trees,” in *Proc. of IFIP Networking*, 2011, pp. 78–91.

[29] E. J. Rosensweig, D. S. Menasche, and J. Kurose, “On the Steady-State of Cache Networks,” in *Proc. of IEEE INFOCOM*, 2013, pp. 887–895.

[30] A. Araldo, M. Mangili, F. Martignon, and D. Rossi, “Cost-aware caching: optimizing cache provisioning and object placement in ICN,” in *Proc. of IEEE Globecom*, 2014, pp. 1108 – 1113.

[31] M. Korupolu and M. Dahlin, “Coordinated placement and replacement for large-scale distributed caches,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 6, pp. 1317–1329, 2002.

[32] S. Borst, V. Gupta, and A. Walid, “Distributed Caching Algorithms for Content Distribution Networks,” in *Proc. of IEEE INFOCOM*, 2010, pp. 1478–1486.

[33] G. Pollatos, O. Telelis, and V. Zissimopoulos, “On the social cost of distributed selfish content replication,” in *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, 2008, pp. 195–206.

[34] E. Jaho, M. Karaliopoulos, and I. Stavrakakis, “Social similarity favors cooperation: the distributed content replication case,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 601–613, 2013.

[35] V. Pacifici and G. Dán, “Selfish Content Replication on Graphs,” in *Proc. of the 23rd International Teletraffic Congress*, 2011, pp. 119–126.

[36] ——, “Distributed Algorithms for Content Allocation in Interconnected Content Distribution Networks,” in *Proc. of IEEE INFOCOM*, 2015, pp. 2362–2370.

[37] ——, “Content-peering Dynamics of Autonomous Caches in a Content-centric Network,” in *Proc. of IEEE INFOCOM*, 2013, pp. 1079 – 1087.