J Netw Syst Manage manuscript No.
(will be inserted by the editor)

End-to-End Resource Management for Federated Delivery of Multimedia Services

Jeroen Famaey · Steven Latré · Tim Wauters · Filip De Turck

the date of receipt and acceptance should be inserted later

Abstract Recently, the Internet has become a popular platform for the delivery of multimedia content. Currently, multimedia services are either offered by Over-the-top (OTT) providers or by access ISPs over a managed IP network. As OTT providers offer their content across the best-effort Internet, they cannot offer any Quality of Service (QoS) guarantees to their users. On the other hand, users of managed multimedia services are limited to the relatively small selection of content offered by their own ISP. This article presents a framework that combines the advantages of both existing approaches, by dynamically setting up federations between the stakeholders involved in the content delivery process. Specifically, the framework provides an automated mechanism to set up end-to-end federations for QoS-aware delivery of multimedia content across the Internet. QoS contracts are automatically negotiated between the content provider, its customers, and the intermediary network domains. Additionally, a federated resource reservation algorithm is presented, which allows the framework to identify the optimal set of stakeholders and resources to include within a federation. Its goal is to minimize delivery costs for the content provider, while satisfying customer QoS requirements. Moreover, the presented framework allows intermediary storage sites to be included in these federations, supporting on-the-fly deployment of content caches along the delivery paths. The algorithm was thoroughly evaluated in order to validate our approach and assess the merits of including intermediary storage sites. The results clearly show the benefits of our method, with delivery cost reductions of up to 80% in the evaluated scenario.

Keywords multimedia service delivery · federation · Service Level Agreement · Quality of Service · contract negotiation

J. Famaey · T. Wauters · F. De Turck
Ghent University – iMinds, Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
Tel: +32-9-3314900
Fax: +32-9-3314899
E-mail: jeroen.famaey@intec.ugent.be

S. Latré
University of Antwerp – iMinds, Middelheimlaan 1, B-2020 Antwerpen, Belgium

1 Introduction

The increasing availability of bandwidth have converted modern communication networks into popular platforms for the delivery of multimedia services, such as Time-Shifted Tele-Vision (TSTV) and Video on Demand (VoD). Many Internet Service Providers (ISPs) have started offering these IPTV-based services (e.g., Verizon, Comcast). Additionally, so called Over-The-Top (OTT) content providers have started offering VoD and live television services to users across the Internet (e.g., Hulu, Netflix). Both approaches have their advantages and disadvantages. As ISPs offer their multimedia services to users within their own managed IP network, they are able to give guarantees concerning the delivered Quality of Service (QoS). However, the operator needs to compose its own content catalogue, acquiring licenses from copyright owners. Moreover, it can serve this content only to a limited number of potential customers. Therefore, the operator will have to choose between limiting costs by offering only recent and popular content, or satisfying all its customers by providing an extensive catalogue that caters all tastes. On the other hand, OTT content providers can offer their services to a potentially huge number of users across the Internet. However, as the current Internet is a best-effort network, they cannot offer any QoS guarantees. Additionally, the content is delivered over the ISP's infrastructure, without it being involved in the control or distribution of the content. This causes the ISP network to become more heavily loaded, while it does not share in the revenue [1].

In this article, we propose an approach to overcome the shortcomings of the two described multimedia content delivery methods. It allows users to consume the wide selection of content offered by the plethora of content providers on the Internet, while additionally supporting end-to-end QoS guarantees. More specifically, a framework is proposed for setting up end-to-end network federations. A federation is defined as a collaboration between network domains, in order to deliver a combined service, or set of services [2]. In the proposed approach, federations are set up between a content provider, a set of intermediary core Internet transit ISPs and an access ISP. This allows the content provider to offer its multimedia content across the Internet to the end-users of the access ISP with guaranteed end-to-end QoS. The intermediary core Internet domains share in the revenues in return for offering bandwidth- and QoS-guaranteed paths through their networks. This article thus considers a Future Internet scenario that supports the provisioning of QoS in the Internet core, which is not possible in the current best-effort Internet. The necessity of providing such guarantees in the Future Internet, has been argued by many Future Internet research efforts [3–6]. The content provider does not directly deal with end-users. For scalability reasons, it instead deals with access ISPs, which act on behalf of a set of end-users. The access ISPs specify their QoS requirements in a Service Level Agreement (SLA), on the terms of which it negotiates with the content provider. The advent of cloud computing has given rise to the possibility of dynamic on-demand reservation of computational and storage resources. This allows content caches to be deployed on-the-fly across the Internet. Additionally, the caching capabilities of Content Delivery Networks (CDNs), which are widely used in the current Internet for the delivery of OTT content, can be leveraged to achieve this. Throughout this article we call such cloud- or CDN-based domains, where content can be dynamically cached by the content provider, storage sites. These storage sites can thus cooperate in the content delivery federations, trading storage resources for part of the revenues. Additionally, in-network deployed content caches can be shared among several access ISPs. This significantly increases their efficiency, as it allows more content to be served without increasing the amount of reserved storage resources. The use of storage sites and cache sharing is a novel aspect of our approach compared to existing end-to-end QoS reserva-

tion mechanisms, which consider only direct QoS-constrained paths between the content provider and consumer.

The process of setting up content delivery federations is facilitated through a mathematical model of the problem. The model defines the stakeholders that can take part in these federations, the end-to-end QoS constraints that need to be satisfied and the cost functions associated with delivering the requested multimedia content. The presented model supports the specification of constraints for multiple QoS parameters simultaneously, as it has been shown that a wide range of different parameters (e.g., bandwidth, delay, jitter, packet loss) affect Quality of Experience [7]. Additionally, this article presents an algorithm for calculating the optimal content delivery paths based on this model. The algorithm minimizes the total delivery cost for the content provider by identifying a set of suitable stakeholders (i.e., transit ISPs and storage sites) that should be included in the delivery federation. Additionally, it calculates the amount of QoS and storage resources that need to be reserved within each of the identified stakeholder domains. Finally, it guarantees the QoS requirements specified in the access ISP SLAs.

A quantitative evaluation, based on a VoD scenario, is performed to validate the presented algorithm and proposed novel content delivery approach. This evaluation has several goals. First, performance and scalability of the algorithm are characterized. Second, the use of intermediary storage sites is compared to an approach that only employs direct end-to-end delivery paths between the content provider and its customers. Third, the merits of cache sharing are evaluated under a variety of conditions. Finally, we explore the potential benefits of deploying caches in the access network in addition to in-network caches.

The remainder of this article is structured as follows. Section 2 describes related work in the context of end-to-end QoS and SLA negotiation. A detailed description of the framework is provided in Section 3. It defines the stakeholders involved in the content delivery federations, as well as the interactions that take place between them. Subsequently, Section 4 goes into more detail about setting up the end-to-end delivery paths and formally models the end-to-end content delivery federation problem solved in this article. Section 5 presents an algorithm to solve the presented problem. Subsequently, its merits are thoroughly validated based on evaluation results of a VoD scenario in Section 6. Section 7 describes how the presented algorithms can be extended to overcome some of the assumptions concerning the content encoding process made in previous sections. Finally, Section 8 concludes the article.

2 Related work

The end-to-end QoS reservation problem can be divided into several sub-problems; finding QoS-constrained shortest paths, negotiating SLAs and managing them. All of these aspects have been actively researched within the network management community. The remainder of this section gives an overview of the most important research efforts on each of these related problems and explains the differences with the work presented in this article. Moreover, an overview of research on the federation of CDNs is provided, as they also employ caching to optimize the delivery of content. However, first the novelty compared to our own previous work is discussed.

The work presented in this article is based on our earlier work [8,9]. The first paper [8] focused on solving a static version of the problem presented here. It assumed the path through the Internet core is fixed and considered only a single content cache shared among all customers. Instead, this article presents an algorithm capable of finding the optimal path through the Internet core. Additionally, it is capable of constructing more complex

delivery trees that consist of multiple content caches, possibly shared among only a subset of all customers. The second paper focused on the SLA management side, and explains how to incorporate the solution of the algorithm into the SLA negotiation process [9].

2.1 Multi-constrained optimal path problem

Finding QoS-constrained cheapest paths through a network is equivalent to the multi-constrained optimal path (MCOP) graph problem [10]. This problem has long been known to be NP-complete [11]. Its goal is to find the shortest (i.e., optimal) path in a graph, subject to multiple edge constraints. Throughout the years, many algorithms and heuristics have been proposed to solve this problem [12,13,11,14–16]. Chen and Nahrstedt proposed an approximation heuristic that attempts to find a feasible solution [12]. However, it cannot guarantee to find a path, even if one exists. The TAMCRA algorithm also finds feasible paths without optimization [13]. If its K parameter is chosen high enough, it has a higher chance of finding the actual optimal path. However, this significantly increases its execution time. The H_MCOP heuristic proposed by Korkmaz and Krunz is a fast approximating heuristic that easily finds feasible solutions if the constraint bounds are loose [11]. However, it often does not yield near-optimal solutions. Several algorithms have also been proposed that do find the optimal, or near-optimal solution. The limited path heuristic is based on an extended version of the Bellman-Ford algorithm [15]. Instead of keeping track of all possible paths from source to destination, it stores only a subset. This reduces its execution time, at the cost of optimality. Liu and Ramakrishnan proposed the optimal A*Prune algorithm [14]. It is an adaptation of the A* searching strategy, combined with a pruning strategy that discards candidate paths that cannot satisfy the constraints. Its runtime is exponential, but a polynomial-time heuristic called BA*Prune is also presented. Finally, Xiao and Boutaba identified several shortcomings with existing MCOP algorithms, making them unsuitable for use in their proposed dynamic service provisioning framework [16]. To alleviate this, they propose a novel fast running heuristic that utilizes a two-step Dijkstra process. First, Dijkstra's shortest path algorithm is employed in reverse in order to determine whether or not there is a feasible path from every node within the network. Second, a normal Dijkstra is used to find the cost-minimizing path that satisfies the constraints. The algorithm has, in the worst case, five times the runtime of Dijkstra's shortest path algorithm. However, it successfully finds the optimal solution in some specific cases where H_MCOP and TAMCRA fail.

2.2 End-to-end Quality of Service

The theoretical work on solving the end-to-end QoS-constrained cheapest path problem is complemented by more practical frameworks for setting up end-to-end QoS-constrained federations. Yan *et al.* presented a multi-agent approach to negotiate QoS for the provisioning of service compositions [17]. Every agent is responsible for provisioning a single service component within the composition. A coordinating agent makes sure the total offered QoS satisfies the requested amount. Pouyllau *et al.* proposed a novel algorithm for setting up end-to-end network federations for the delivery of QoS-constrained services [10]. They assume the set of candidate paths has already been found, using an existing MCOP algorithm. They then formulate the problem of selecting the optimal QoS-constrained path as a game theoretic problem. More recently, they have proposed a reinforcement learning algorithm, based on Q-learning and Markov Decision Processes, to solve the previously formulated game [6].

Their goal is to maximize long-term revenues, while performing real-time treatment of customers' requests. Amigo *et al.* focus on the economics of end-to-end QoS-aware federations [1]. They argue that the current service business model of the Internet is unbalanced, where not all intermediary network domains receive their fair share of revenues. To alleviate these economic concerns, they propose an end-to-end bandwidth allocation framework. It allows transit network domains to collaborate in order to offer bandwidth guaranteed end-to-end pipes through the Internet.

In addition to algorithms that find QoS-constrained optimal paths, there is also a need for QoS-aware routing protocols in order to support end-to-end QoS on the Internet. Several evolutionary approaches have been proposed to support end-to-end QoS on top of the current best-effort Internet. Kumar *et al.* presented the Alliance network model [3]. It allows interconnected Autonomous Systems (AS) to form an alliance or federation, which enables optimal inter-domain path selection and QoS guarantees. Additionally, it is compatible with the Border Gateway Protocol (BGP) and can thus co-exist with the current best-effort Internet. Xiangjiang *et al.* presented a similar approach, also based on BGP [4].

Our work differs from existing work on end-to-end QoS provisioning, in the sense that we aim to solve an extension of the MCOP problem. In addition to finding QoS-constrained shortest paths, our goal is to set up end-to-end delivery federations that additionally include intermediary content caches. As such, existing MCOP algorithms cannot be directly applied to this extended problem. Nevertheless, they are incorporated into our novel algorithm to solve a subset of the extended problem.

Recently, Balasubramaniam *et al.* proposed an integrated architecture combining service lifecycle management and dynamic end-to-end routing [18]. The proposed management framework is inspired by biological systems, and aims to autonomously configure services and the underlying routing system in order to guarantee the ever-changing end-to-end QoS requirements of a large number of heterogeneous services. Although their research has similar goals to ours, they focus on service lifecycle management and dynamic distributed routing. In contrast, our work centers around long-term cost minimization through intelligent federation composition and resource management.

2.3 SLA negotiation and management

In order to successfully set up a federation, an agreement should be negotiated between the participants. To achieve this, SLA negotiation protocols can be employed. Several frameworks and architectures have been proposed to support SLA negotiation between federation partners. Yuanming *et al.* developed a framework for the negotiation of SLAs between service providers, network operators and content providers [19]. More recently, the SLA-based SERViceable Metacomputing Environment (SERVME) was proposed [20]. It consists of a framework and accompanying SLA model, which guide the SLA negotiation process, match providers based on QoS requirements and perform on-demand resource provisioning. In addition to frameworks that support the negotiation and management of SLAs, several protocols that allow the actual SLA terms to be negotiated have been proposed. The Web Services Agreement Specification (WS-Agreement) is a Web Services protocol for establishing agreement between two parties, such as a service provider and consumer [21]. It consists of an XML-based language for specifying SLAs, as well as a protocol for negotiating its terms [22]. Hudert *et al.* extended these ideas with a framework built around WS-Agreement that supports multilateral in addition to bilateral negotiations [23]. Hasselmeyer *et al.* proposed a Discrete-Offer-Protocol that allows the service provider to make a single

offer, which the consumer can accept or reject [24]. More recently they proposed a more elaborate protocol [25]. It supports multi-round negotiations, as well as re-negotiating the terms of an SLA already in place as the requirements of participants change. The actual SLA negotiation protocol is outside the scope of our work. Instead we focus on determining the costs associated with complying to the terms of an SLA. This calculated cost can then be taken into account during the subsequent negotiation process.

2.4 Content Delivery Networks

CDNs are a popular method for delivering content in today's Internet [26]. They are large-scale distributed systems that augment the Internet's best effort delivery mechanism through caching, as well as intelligent content replication and server selection. Specifically, CDNs aim to improve QoS by delivering content from the optimal location, in terms of for example geographical distance, end-to-end delay or available bandwidth. This is accomplished through a wide range of techniques, such as caching (passive or proactive), active end-to-end measurements and/or estimation of the user's geographical location.

Traditionally, CDNs operate solely on self-obtained information and metrics, which is often inaccurate and gives only an estimate of end-to-end performance rather than specifics of the traversed networks and links [27,28]. In line with the vision presented in this article, there has been a recent push towards collaborative techniques between CDNs and ISPs, giving the latter the opportunity to steer the CDN server selection process based on more accurate information from its own network. Frank *et al.* proposed Content-aware Traffic Engineering (CaTE) [28]. The framework leverages the location diversity offered by distributed CDNs, influencing the delivery path of content by selecting an appropriate server. Results showed a significant reduction in network-wide traffic as well as end-to-end delay. Liu *et al.* propose the deployment of a global video control plane to coordinate the CDN and server selection process [29]. Such a plane copes with the temporal and spatial variability in CDN performance by dynamically changing the selected CDN and/or server from which to stream video content. This has both a direct performance benefit, as well as increased flexibility in instrumenting fine-grained policies by the content provider.

Moreover, several standardization efforts have been launched to facilitate the cooperation among CDNs [30,31]. CDN-interconnection (CDNi) allows CDNs to collaborate in delivering content; increasing geographic range, facilitating capability sharing, and supporting inter-CDN load balancing. The goal of these efforts is to standardize control, routing, and logging interfaces.

In many ways, CDNs are similar to the vision proposed in this article. They also aim to improve delivery quality and reduce costs through the deployment of distributed caches. Additionally, there has been much recent effort to facilitate the collaboration between CDNs and ISPs and among CDNs. Nevertheless, traditional CDNs focus on improving performance on top of the current best effort Internet. As such, providing strong QoS guarantees is outside the scope of such solutions. In contrast, the Future Internet content delivery framework proposed in this article aims to provide such delivery guarantees through the reservation of capabilities in ISP networks. Although such systems are not deployed in the Internet today, several research efforts exist that can be leveraged to facilitate this. They include Software Defined Networking (SDN) [32], network virtualization [33], and inter-domain traffic engineering [34]. We view the increased popularity of such paradigms, both in academia and industry, as an indication of an evolution towards the future deployment of QoS reservation mechanisms.

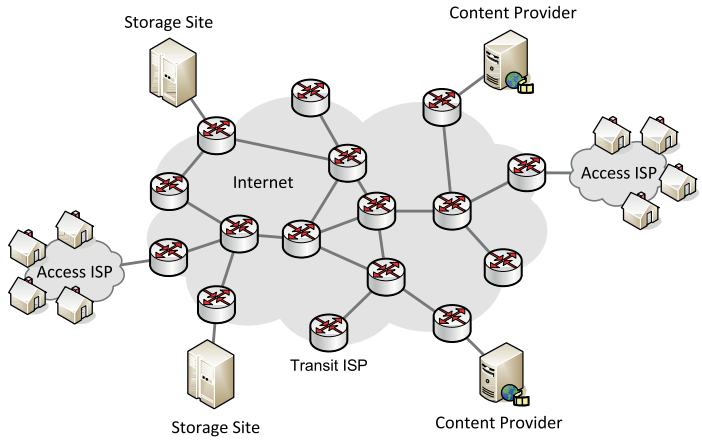


Fig. 1: An overview of the stakeholders involved in the end-to-end content delivery process: content providers, storage sites, access ISPs and transit ISPs

However, we believe that our approach is, to some extent, complementary to CDN-based content delivery. The proposed ecosystem includes a storage site stakeholder, which takes part in content delivery federations to provide storage resources for deploying dynamic caches. Such a role can be easily adopted by current CDN providers, either leasing out storage resources directly to content providers or through the use of their self-managed caching infrastructure.

3 Federated content delivery framework

The goal of the envisioned framework is to facilitate the end-to-end delivery of multimedia content across the Internet. To achieve this, it allows content providers and access ISPs to set up federations with transit ISPs and storage sites along the end-to-end path connecting them. Cooperating in a federation holds advantages for all involved stakeholders. On one hand, it allows the content provider to guarantee QoS across the Internet. On the other, it entitles the intermediary network domains to a share of the content provider's revenue. The remainder of this section identifies the different stakeholders involved in these federations, describes how they interact to set them up, and briefly discusses the potential revenue gain for each stakeholder by adopting the presented approach. Sections 4 and 5 further elaborate on the algorithmic details of the federation set-up process.

3.1 Stakeholders

Figure 1 positions the different stakeholders throughout the network. There are four types of stakeholders involved in the content delivery federations: content providers, access ISPs, transit ISPs, and storage sites. The content providers and storage sites are positioned at the edge of the Internet, connected to the remainder of the network through a single transit ISP. The *content provider* locally hosts a set of multimedia content items. It aims to sell these to interested access ISPs. Traditionally, the *access ISP* provided Internet access to a set of

end-users. Nowadays they often offer novel multimedia services, including TSTV and VoD. Our framework builds on this and considers the access ISPs the direct customers of the content provider. In line with current advances, the access ISP is assumed to offer multimedia services to the end-users. However, in contrast to current approaches, it does not manage its own content catalogue, but rather obtains content via one or more content providers, over the Internet. The Internet core consists of many *transit ISPs*, together forming a network of networks. They connect the different edge domains to the Internet, and are responsible for routing network traffic from source to destination. In the current Internet, end-to-end routing is static and best-effort. However, future Internet research has advanced the idea of on-demand end-to-end QoS provisioning within the Internet core [3,4,6]. As such, transit ISPs can be included in the end-to-end federations, providing QoS guarantees in return for a share of the content provider's revenue. A set of *storage sites* is spread across the Internet. They enable on-the-fly provisioning of storage resources. This allows content providers to dynamically deploy content caches across the Internet. Possible storage site providers include cloud storage providers as well as CDNs. The former give full control to the content provider, which leases low-level storage resources on which it deploys, configures, and manages its own caching components. In the latter case, the content provider outsources cache control to the CDN, which usually manages its own caching infrastructure. The interactions described below assume full control lies with the content provider. Nevertheless, control could alternatively be distributed among different stakeholders, supporting the incorporation of CDNs within the proposed content delivery ecosystem.

In summary, the presented framework thus combines characteristics and advantages of the two described existing multimedia content delivery approaches. First, it allows content providers to offer their content to end-users across the Internet, as is the case in the OTT scenario. However, in contrast to OTT content delivery, access ISPs are still involved in the delivery process and can share in the revenue. Thus allowing access ISPs to offer a plethora of multimedia services, as they currently do, without needing to maintain their own content catalogue. Additionally, by including transit ISPs and storage sites in the delivery federations, QoS guarantees can be offered and transmission costs can be decreased, further reducing disadvantages of OTT content delivery.

3.2 Interactions

The stakeholders involved in the content delivery process interact in several ways to set up QoS-guaranteed end-to-end paths across the Internet. The federation set-up process is initiated by the access ISP when it decides it wants to offer the content of a specific content provider to the end-users connected to its network. Figure 2 depicts the negotiation process from the content provider to an access ISP in detail. First, the ISP requests the list of quality levels offered by the content provider. The content provider replies with information about each quality level. Depending on the type of multimedia content the offered information may differ. It could include for example spatial resolution, temporal resolution, and bit-rate. The access ISP can use this plethora of information to determine which quality levels its end-users will be interested in (based on their device capabilities and expectations). From the content provider's perspective the bit-rate is the important attribute, as it determines the amount of bandwidth resources that need to be reserved and consequently the cost associated with delivering the content. Subsequently, the access ISP requests an initial price offer for delivering content with a specific quality and specific QoS requirements (e.g., delay, jitter, packet loss, availability). The content provider uses its price models and associated

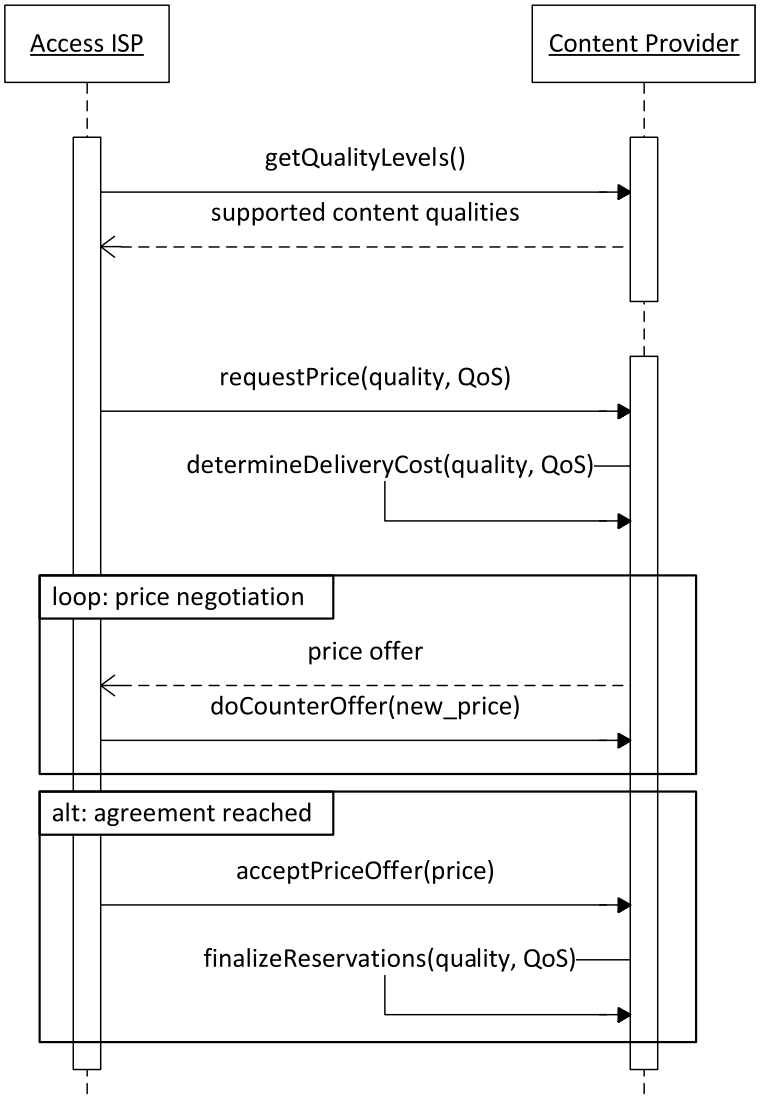


Fig. 2: A sequence diagram detailing the negotiation process between the content provider and an access ISP customer

algorithms to determine the costs associated with delivering its content to the access ISP. To calculate this cost, it additionally needs to find a cost-minimizing end-to-end path and identify the network domains along this path. This is a complex problem in itself for which we offer a solution in Sections 4 and 5. The content provider can then formulate an initial price offer to the ISP, based on its calculated delivery costs and the expected profit. This negotiation can end in either agreement or disagreement. If an agreement is reached, the SLA negotiation process is finalized and the ISP can start offering the content provider's multimedia content to its end-users. This finalized SLA contains the negotiated price, con-

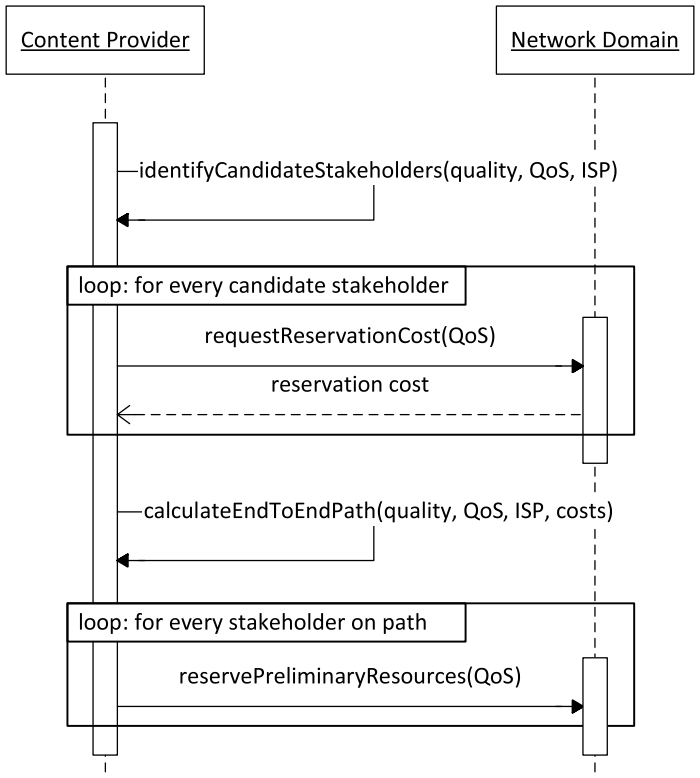


Fig. 3: A sequence diagram detailing the interactions involved in finding an end-to-end delivery path from the content provider to an access ISP

tent quality, QoS requirements and the time-frame over which it is valid. The negotiated SLA is only valid for a certain period of time to take into account changes in delivery costs. After the time-frame has expired, a new SLA should be negotiated taking into account current parameters and costs. If no agreement is reached during the negotiation procedure, the federation is not finalized. It is then up to the access ISP to start again, either by contacting another content provider or requesting less stringent QoS.

As previously stated, the content provider needs to determine the costs associated with delivering its content across the Internet towards its federated access ISPs. Additionally, it needs to reserve the necessary resources in order to satisfy its obligations captured in the negotiated SLA. This introduces a set of additional interactions between the content provider on one hand and the transit ISPs and storage sites along the end-to-end path on the other hand. Figure 3 depicts the interactions involved in finding an end-to-end delivery path between the content provider and an access ISP. In order to calculate the costs involved in delivering its content to a specific access ISP, the content provider first needs to identify the network domains the content will pass through. These domains can either be transit ISPs or storage sites. In case of the former, the content provider requests the cost per bandwidth unit for reserving a path through the domain with the requested QoS. In case of the latter, it asks the cost for reserving storage resources. This cost information is then employed by the content provider to calculate the cost-minimizing end-to-end path to the access ISP

that satisfies the requested QoS. Finally, it performs a preliminary lock on the necessary resources within the domains along this path, to make sure it can satisfy the offer that will be made to the access ISP. These preliminary reservations are finalized as soon as the content provider and access ISP come to an agreement. If they do not, the locks are released.

3.3 Revenue model

The envisioned federated content delivery approach has several economic benefits for the identified stakeholders, compared to the currently popular OTT and ISP-based delivery methods. In current OTT scenarios, only the service and content providers actually benefit economically. The customer pays fees (e.g., subscription fee or per-item rental fee) directly to the service provider, which in turn licenses the content from the content provider. Although such services consume considerable bandwidth from ISP networks, they do not receive a share of the revenues [1]. In contrast, access ISPs do receive revenues in the ISP-based delivery approach. However, this comes at the cost of considerable infrastructure investment and maintenance costs, as well as content licensing fees. The revenue distribution in these traditional delivery methods is thus unbalanced, favouring those that offer the content and services, rather than those that transport them.

The envisioned federated delivery approach would result in a more balanced revenue model, where all involved stakeholders benefit financially or otherwise. Specifically, the ISPs, responsible for transporting content, will receive part of the revenue and will thus be able to recuperate their bandwidth costs, in return for providing delivery guarantees. Content providers will be able to offer more diverse delivery classes to their customers, increasing user satisfaction and justifying more costly subscription models. Today's OTT service providers (e.g., Netflix, Hulu), which often act as content brokers by licensing content from a wide range of content providers, will still be able to fill that role in the proposed approach, receiving the same advantages as the content provider stakeholder. Moreover, end-users will be able to enjoy a wider range of quality guarantees. Finally, storage providers (e.g., cloud storage providers or CDNs) will fill a similar business role as transit ISPs, offering added-value delivery functionality in return for a share of the revenues.

4 End-to-end content delivery federations

The set of stakeholders involved in the end-to-end delivery of multimedia content from the content provider to one of its access ISP customers takes the form of an end-to-end path through the Internet, consisting of transit ISPs and storage sites. This section elaborates on the problem of finding such an end-to-end path between a content provider and its access ISP customers. Every path should satisfy the QoS constraints requested by the associated access ISP. Additionally, of all possible combinations of paths, the cost minimizing set should be selected. First, the problem domain is mathematically modelled. Second, a formal problem formulation is presented. Section 5 presents a heuristic to solve this mathematical problem.

4.1 Notations & assumptions

Let us consider an Internet topology, consisting of an interconnected set of ISPs $\boldsymbol{I}$. Every ISP $i \in \boldsymbol{I}$ is connected to a set of neighbours $\boldsymbol{I}_i$. There are two types of ISPs: transit ISPs $\boldsymbol{T}$ and

access ISPs $\boldsymbol{A}$. Additionally, there are several types of edge network domains involved in the content delivery process: a set of content providers $\boldsymbol{P}$ and a set of storage sites $\boldsymbol{S}$. Every content provider and storage site $e \in \boldsymbol{P} \cup \boldsymbol{S}$ is connected to the Internet through its gateway $t_e \in \boldsymbol{T}$.

In the considered scenario, it is possible to reserve QoS-guaranteed paths through the Internet core. To this end, every transit ISP $t \in \boldsymbol{T}$ offers a set of QoS classes $\boldsymbol{C}_t \subseteq \boldsymbol{C}$. A QoS class $c \in \boldsymbol{C}$ offers a QoS guarantee $\gamma_{c,q}$ for every QoS parameter $q \in \boldsymbol{Q}$ (e.g., delay, packet loss, jitter, availability), and has an associated transmission cost θ_c per unit of content. Concretely, a QoS guarantee specifies a constraint on the associated QoS parameter (e.g., delay $\leq$ 100ms, availability $\geq$ 99.999%). Different QoS parameters are aggregated in different ways. For example, the total end-to-end delay is the sum of the individual link delays, while the total availability is the product of all link availabilities. As such, we define the end-to-end QoS aggregation operator of a QoS parameter $q \in \boldsymbol{Q}$ as $\oplus_q$. It calculates the aggregated value of the QoS parameter based on the individual link values. Similarly, the satisfaction operator to determine if a QoS value satisfies a requirement depends on the QoS parameter. For example, delay is satisfied if the guaranteed value is below the constraint, while availability is satisfied if it is above. As such, we define the satisfaction operator of a QoS parameter q as $\prec_q$. Specifically, QoS value a satisfies constraint b if $a \prec_q b$ equals true.

Every content provider $p \in \boldsymbol{P}$ offers a content catalogue, available in a set of bit rates $\boldsymbol{B}_p$. The content catalogue is additionally characterised by some statistical information, which is necessary to calculate the total delivery cost. This information consists of the weighted average content duration δ_p (with the weight proportional to the item's popularity), the number of items in the catalogue χ_p and for every bit rate $b \in \boldsymbol{B}_p$ a cumulative popularity distribution function $\Phi_{p,b}(\cdot)$. This function takes as input an integer x, and returns as output the probability that an end-user requests one of the x most popular items in the content catalogue. For example, $\Phi_{p,b}(100) = 0.9$, means that 90% of all requests for content with bit rate b received by content provider p are for its 100 most popular items.

The negotiation of SLAs is governed by a set of delivery requests $\boldsymbol{R}$. These requests stipulate the demands of access ISPs pertaining to the content they want to receive from a specific content provider. Note that these requests are not requests for a single content item, but rather requests for an end-to-end delivery path, over which multiple content items can be sent with specific QoS guarantees. The requests associated with a content provider $p \in \boldsymbol{P}$ are defined as $\boldsymbol{R}_p \subseteq \boldsymbol{R}$, while the set of requests originating from an access ISP $a \in \boldsymbol{A}$ is defined as $\boldsymbol{R}_a \subseteq \boldsymbol{R}$. A request $r \in \boldsymbol{R}$ originates from an access ISP $a_r \in \boldsymbol{A}$, which sends it to a content provider $p_r \in \boldsymbol{P}$. Furthermore, it contains a requested bit rate $b_r \in \boldsymbol{B}_{p_r}$, an expected number of simultaneous delivered content items ρ_r and an end-to-end QoS constraint $\gamma_{r,q}$ for every QoS parameter $q \in \boldsymbol{Q}$.

Storage sites support the on-demand reservation of storage resources. This allows the content providers to deploy dynamic caches throughout the Internet. Several caches, associated with different requests, may be deployed within a single storage site. In order to support cache sharing, a single cache may also be associated with multiple requests. Note that caches can also be deployed within access ISP domains. However, their available resources are expected to be limited compared to those of dedicated storage sites, and a small, static cache is thus associated with them. For genericity, we also assume every content provider hosts a content cache that stores its entire content catalogue. A set of content caches $\boldsymbol{O}$ is deployed within domains across the Internet. A content cache $o \in \boldsymbol{O}$ is characterised by the domain $d_o \in \boldsymbol{S} \cup \boldsymbol{A} \cup \boldsymbol{P}$ in which it is deployed, its cache size σ_o and the set of requests $\boldsymbol{R}_o \subseteq \boldsymbol{R}$ for which it is used. It is assumed that all requests $r \in \boldsymbol{R}_o$ request the same bit-rate $b_o \in \boldsymbol{B}$ and target the same content provider $p_o \in \boldsymbol{P}$. A storage cost λ_s and processing cost φ_s are asso-

ciated with every storage site $s \in \boldsymbol{S}$ per unit of content. The storage cost is paid for cached content only, while the processing cost is paid for every content request that it serves. It is assumed there are no costs associated with the caches deployed in the access ISP and content provider domains. As such, the incentive for the access ISP to host a cache is a reduction in the price it pays to the content provider rather than an actual share of the revenues.

For every request $r \in \boldsymbol{R}$, an associated end-to-end delivery path $\Pi_r = \langle o_1, ..., o_n \rangle$ is set up, with $o_i \in \boldsymbol{O}$ for $i \in [1,n]$, $d_{o_1} = p_r$, $d_{o_n} = a_r$ and if $n > 2$, $d_{o_i} \in \boldsymbol{S}$ for $i \in [2,n-1]$. In other words, the path consists of a set of content caches, with the source cache deployed within the content provider domain, the target cache within the access ISP domain and any remaining intermediary caches within storage sites. The successor and predecessor of $o \in \Pi_r$ along a path Π_r are respectively defined as o_r^+ and o_r^-. When an end-users requests a content item, the associated content request is sent to the content caches along this path in reverse order (i.e., first to o_n, then to o_{n-1}, ...). The request is forwarded until a cache is encountered that locally stores the requested content item. As the cache associated with the content provider stores all content, every request is eventually answered. The content itself is not sent via the end-to-end delivery path, but via a direct path through the Internet core. As such, a core Internet path $\pi_{o,r} = \langle g_1, ..., g_m \rangle$ is associated with every cache $o \in \Pi_r \setminus \{o_n\}$, with $g_1 = t_{d_o}$, $g_n \in \boldsymbol{I}_{a_r}$ and $g_i \in \boldsymbol{T}$ for $i \in [1,m]$. A core Internet path thus consists of only transit ISPs. Its first transit ISP is the gateway of d_o, while its last is a neighbour of a_r. For every transit ISP $t \in \pi$ along the core Internet path $\pi_{o,r}$, a QoS class $c_{t,r} \in \boldsymbol{C}_t$ is reserved.

In summary, Table 1 lists the symbols introduced throughout this section.

4.2 Problem formulation

As stated, the problem consists of finding the cost minimizing end-to-end delivery path Π_r for every request $r \in \boldsymbol{R}$. Additionally, for every $o \in \Pi_r \setminus \{o_n\}$ a path $\pi_{o,r}$ through the Internet core from $d_o \in \boldsymbol{S} \cup \{p_r\}$ to $a_r \in \boldsymbol{A}$ that satisfies the request's QoS constraints needs to be found. Formally, $\pi_{o,r}$ should therefore satisfy the following constraints:

$$\forall r \in \boldsymbol{R}, \forall q \in \boldsymbol{Q}, \forall o \in \Pi_r \setminus \{o_n\} : \bigoplus_{t \in \pi_{o,r}} \gamma_{c_{t,r},q} \prec \gamma_{r,q} \tag{1}$$

The total cost of Π_r consists of three components: transmission cost, storage cost and processing cost. The storage and processing cost are related to the storage sites, while the transmission cost is related to the reservation of QoS in the transit ISP domains. The total storage cost of all content caches is calculated as follows:

$$\Delta = \sum_{o \in \boldsymbol{O}} \sigma_o \times \lambda_{s_o} \times b_o \times \delta_{p_o} \tag{2}$$

The processing and transmission costs both depend on the number of content items that are served from a specific cache. Additionally, these costs are influenced by other content caches on the path towards the access ISPs. More specifically, the cache deployed nearest the access ISP will serve the most popular content, the second nearest then serves the most popular content not served by the first, and so on. However, as content caches can be shared, they may belong to multiple end-to-end delivery paths and thus have multiple child caches. This makes it difficult to determine the exact number of popular items that are served downstream. As such, we use the minimum of all direct child caches as a lower bound. First, we calculate the aggregated cache size of a content cache $o \in \boldsymbol{O}$, which is defined as the lower

Table 1: The list of symbols used throughout this article

Symbol	Explanation
$\oplus_q$	aggregation operator of QoS parameter $q \in \boldsymbol{Q}$
$\prec_q$	satisfaction operator of $q \in \boldsymbol{Q}$
$\boldsymbol{A}$	access ISPs
$a_r \in \boldsymbol{A}$	source ISP of request $r \in \boldsymbol{R}_a$
$\boldsymbol{B}_p$	bit rates offered by $p \in \boldsymbol{P}$
$b_o \in \boldsymbol{B}_{p_o}$	bit rate of content stored in content cache $o \in \boldsymbol{O}$
$b_r \in \boldsymbol{B}_{p_r}$	bit rate associated with request $r \in \boldsymbol{R}$
$\boldsymbol{C}$	QoS classes
$\boldsymbol{C}_t \subseteq \boldsymbol{C}$	QoS classes offered by $t \in \boldsymbol{T}$
$c_{t,r} \in \boldsymbol{C}_t$	QoS class reserved in $t \in \boldsymbol{T}$ for request $r \in \boldsymbol{R}$
χ_p	number of items in the content catalogue of $p \in \boldsymbol{P}$
$d_o \in \boldsymbol{S} \cup \boldsymbol{A} \cup \boldsymbol{P}$	domain where content cache $o \in \boldsymbol{O}$ is deployed
δ_p	average duration of content offered by $p \in \boldsymbol{P}$
$\gamma_{c,q}$	QoS guarantee of QoS parameter $q \in \boldsymbol{Q}$ for QoS class $c \in \boldsymbol{C}$
$\gamma_{r,q}$	QoS constraint of QoS parameter $q \in \boldsymbol{Q}$ for request $r \in \boldsymbol{R}$
$\boldsymbol{I}$	transit and access ISPs
$\boldsymbol{I}_i \subseteq \boldsymbol{I}$	the neighbours of $i \in \boldsymbol{I}$
λ_s	storage cost associated with domain $s \in \boldsymbol{S}$
$\boldsymbol{O}$	content caches
$o_r^+ \in \Pi_r$	successor of $o \in \boldsymbol{O}$ along the end-to-end path Π_r of $r \in \boldsymbol{R}$
$o_r^- \in \Pi_r$	predecessor of $o \in \boldsymbol{O}$ along the end-to-end path Π_r of $r \in \boldsymbol{R}$
$\boldsymbol{P}$	content providers
$p_o \in \boldsymbol{P}$	content provider associated with content cache $o \in \boldsymbol{O}$
$p_r \in \boldsymbol{P}$	target content provider of request $r \in \boldsymbol{R}_p$
$\Phi_{p,b}(\cdot)$	cumulative popularity distribution of $p \in \boldsymbol{P}$ for $b \in \boldsymbol{B}_p$
φ_s	processing cost on $s \in \boldsymbol{S}$
$\Pi_r \subseteq O$	end-to-end path associated with $r \in \boldsymbol{R}$
$\pi_{o,r}$	core Internet path from $d_o \in \boldsymbol{P} \cup \boldsymbol{S}$ to $a_r \in \boldsymbol{A}$
$\boldsymbol{Q}$	QoS parameters
$\boldsymbol{R}$	delivery requests
$\boldsymbol{R}_a \subseteq R$	requests for $a \in \boldsymbol{A}$
$\boldsymbol{R}_o \subseteq R$	delivery requests for which content will be served from content cache $o \in \boldsymbol{O}$
$\boldsymbol{R}_p \subseteq R$	requests for $p \in \boldsymbol{P}$
ρ_r	expected number of simultaneous delivered content items for $r \in \boldsymbol{R}$
$\boldsymbol{S}$	storage sites
σ_o	cache size of content cache $o \in \boldsymbol{O}$
$\boldsymbol{T}$	transit ISPs
$\boldsymbol{t}_e \subseteq \boldsymbol{T}$	gateway of $e \in \boldsymbol{P} \cup \boldsymbol{S}$
θ_c	reservation cost associated with QoS class $c \in \boldsymbol{C}$

bound on the number of content items that are served by o or any of the caches on the delivery paths from o towards the access ISPs it serves. The aggregated cache size of o is then defined as follows:

$$\sigma_o^{\text{aggr}} = \sigma_o + \min_{r \in \boldsymbol{R}_o} \sigma_{o_r^+}^{aggr} \tag{3}$$

If $d_o \in \boldsymbol{A}$, this equation is trivially reduced to $\sigma_o^{\text{aggr}} = \sigma_o$, as in such a case o has no successors. For any request $r \in \boldsymbol{R}$, the cumulative popularity distribution function $\Phi_{p_r,b_r}(\cdot)$ and the aggregated cache size can then be used to calculate the percentage of requests answered by every $o \in \Pi_r$:

$$\Phi_{p_r,b_r}(\sigma_o^{\text{aggr}}) - \Phi_{p_r,b_r}\left(\sigma_{o_r^+}^{\text{aggr}}\right) \tag{4}$$

This allows us to calculate the total processing cost as follows:

$$\Psi = \sum_{o \in \boldsymbol{O}} \sum_{r \in \boldsymbol{R}_o} \left(\Phi_{p_r,b_r}(\sigma_o^{\text{aggr}}) - \Phi_{p_r,b_r}\left(\sigma_{o_r^+}^{\text{aggr}}\right) \right) \times b_r \times \varphi_{d_o} \times \rho_r \tag{5}$$

And finally the transmission cost:

$$\Theta = \sum_{o \in \boldsymbol{O}} \sum_{r \in \boldsymbol{R}_o} \sum_{t \in \pi_{o,r}} \left(\Phi_{p_r,b_r}(\sigma_o^{\text{aggr}}) - \Phi_{p_r,b_r}\left(\sigma_{o_r^+}^{\text{aggr}}\right) \right) \times b_r \times \theta_{c_t,r} \times \rho_r \tag{6}$$

In summary, the goal of the content delivery federation problem is to minimize $\Delta + \Psi + \Theta$, while satisfying the constraints specified in Eq. 1.

5 End-to-end resource reservation algorithm

This section presents a heuristic to solve the problem described in Section 4.2. The sub-problem of finding QoS constrained paths through the Internet core is equivalent to the MCOP problem [10]. This has been shown to be NP-complete [11]. Therefore, the federated content delivery problem presented in this article is also NP-complete. We propose a heuristic that starts from a sub-optimal feasible solution and iteratively improves it. The initial solution consists of the minimum cost QoS-constrained Internet core paths directly from the content provider to the access ISP. Subsequently, storage sites are iteratively included in the end-to-end paths and cache sharing opportunities are identified in order to further reduce the total delivery cost. Figure 4 depicts the steps of the devised algorithm in more detail. The sections in which the steps are described are denoted in parentheses. The remainder of this section formally describes the different steps of the heuristic, and how the input parameters required by the algorithm can be statistically estimated. The section is concluded with a discussion of the worst-case time complexity of the heuristic's steps.

5.1 Finding QoS-constrained core paths

Both the initial set-up step and the subsequent iterative improvement steps of the presented heuristic rely on finding QoS-constrained minimum cost paths through the Internet core. As such, we consider this sub-problem first before focusing on the actual steps of the heuristic. As stated, this problem is equivalent to the MCOP problem, which finds the shortest path in a graph subject to one or more constraints. Several optimal algorithms and sub-optimal heuristics have been proposed in literature to solve this problem [35,36]. In our implementation we chose the *A*Dijkstra* variant of the *A*Prune* algorithm [14]. It is capable of both finding the optimal solution (in exponential time) or an approximation (in polynomial time). Nevertheless, any other algorithm that solves the MCOP problem could be used instead.

To find the minimum cost core Internet path $\pi_{o,r}$ for a content cache $o \in \boldsymbol{O}$ and a request $r \in \boldsymbol{R}$, the MCOP algorithm takes as input a source domain $t_{d_o} \in \boldsymbol{T}$ (i.e., the gateway of the domain in which o is deployed), a target domain $a_r \in \boldsymbol{A}$, a QoS constraints $\gamma_{r,q}$ for every $q \in \boldsymbol{Q}$ and a graph G representing the network topology. The set of vertices of G equals the set of ISPs $\boldsymbol{I}$. Every $t \in \boldsymbol{T}$ has one outgoing edge for every QoS class $c \in \boldsymbol{C}_t$ to all of its neighbours $n \in \boldsymbol{I}_t$. Every $a \in \boldsymbol{A}$ thus only has incoming edges. The cost of an edge equals the cost θ_c of the associated QoS class c, while the weights equal the QoS values $\gamma_{c,q}$ of c for all QoS parameters $q \in \boldsymbol{Q}$. As a solution, the MCOP algorithm returns a path $\pi_{o,r}$ of

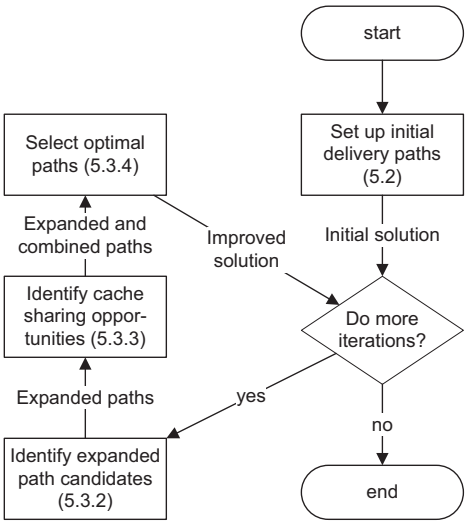


Fig. 4: Flowchart depicting the steps and flow of the resource reservation algorithm; The sections in which the steps are described are denoted in parentheses

vertices and edges. This path is guaranteed to satisfy the constraints specified in Eq. 1 for r. If the algorithm is optimal, the path is also guaranteed to have the minimum aggregated cost of all feasible paths. As every edge in G is associated with exactly one QoS class $c \in \boldsymbol{C}$, the reserved QoS classes $c_{t,r}$ for every $t \in \pi_{o,r}$ can be unambiguously derived from the edges selected by the algorithm.

5.2 Setting up the initial delivery paths

As the federated content delivery problem is NP-complete, it is impractical to expect to find the optimal solution within a feasible time-frame. Therefore, we propose a heuristic that iteratively improves its solution. This section describes the first iteration, while the subsequent iterative improvement process is discussed in the next section. To be able to quickly set-up content delivery federations, this initial step has a considerably lower computational complexity than the improvement steps. As a trade-off, the initial solution is less cost-efficient. However, the initial federation can be set-up using the algorithm's initial solution and refinements can be done over time as better solutions are discovered.

For every $r \in \boldsymbol{R}$, the initial iteration creates a trivial end-to-end path $\Pi_r = \langle o_{p_r}, o_{a_r} \rangle$, consisting of the content caches deployed in the content provider p_r and access ISP a_r domains associated with r. Using the method described above, the minimum-cost QoS constrained core Internet path $\pi_{o_{pr},r}$ from p_r to a_r is then calculated. In this initial solution, all content is thus requested directly from the content provider.

5.3 Iteratively improving the delivery paths

The initial solution calculated above does not include any storage sites. Including storage sites, and deploying dynamic content caches within them, can nevertheless significantly

reduce transmission costs. The improvement process expands existing end-to-end paths by adding intermediary content caches. As previously stated, cache sharing allows multiple access ISPs to benefit from a single cache, while sharing the storage cost among them. This allows the total delivery cost of the solution to be further reduced. As such, the improvement process additionally identifies cache sharing opportunities and deploys shared caches when appropriate. Note that cache sharing can only be done among requests that share the same content provider $p \in \boldsymbol{P}$ and bit-rate $b \in \boldsymbol{B}_p$. As such, the remainder of this section assumes all requests are directed at the same content provider p and request the same bit-rate b. The steps can then be repeated in the same fashion for other content providers and bit-rates to solve the entire problem.

Iteration (i) takes as input an end-to-end path $\Pi_r^{(i-1)}$ for every $r \in \boldsymbol{R}$ calculated during iteration $(i-1)$. The goal is to find an end-to-end path $\Pi_r^{(i)}$ for every $r \in \boldsymbol{R}$, while satisfying the following inequality:

$$\Delta^{(i)} + \Psi^{(i)} + \Theta^{(i)} \leq \Delta^{(i-1)} + \Psi^{(i-1)} + \Theta^{(i-1)} \tag{7}$$

In other words, the total cost of the solution found in iteration (i) should be less than or equal to the total cost of the solution found in iteration $(i-1)$. The improvement process consists of several steps. First, for every path, the set of storage sites that could potentially reduce its total cost, is identified. Second, cache sharing opportunities are identified. Third, from all the candidates, the optimal set of delivery paths is selected. In order to calculate the optimal cost of an end-to-end path, we need to be able to determine the optimal cache size of all caches along the path. As such, the remainder of this section first presents a method for determining the optimal cache size. Subsequently, the three sub-steps are discussed in more detail.

5.3.1 Determining optimal cache sizes

In this section we present a method for calculating the optimal size of the content caches in an end-to-end path Π_r. As caches can be shared, the optimal cache sizes should be calculated simultaneously for all paths that share at least one cache. The presented method thus calculates the optimal cache sizes, given a set of overlapping end-to-end paths $\{\Pi_r\}_{r \in \boldsymbol{R}}$. The problem can be formulated as a Linear Programming (LP) problem, which can be solved with standard LP solving algorithms such as the simplex method [37]. An LP formulation consists of decision variables, constraints and an objective function. The decision variables represent the unknowns, which in this case are the cache sizes. As such, a decision variable $\sigma_o \in [0, \chi_p]$ is associated with every content cache $o \in \{\Pi_r\}_{r \in \boldsymbol{R}}$. As the cache sizes of content providers and access ISPs are assumed to be static, those decision variables are set to a predefined constant value. The objective function minimizes the total cost (i.e., $\Delta + \Psi + \Theta$). As cache size is an integer variable, the problem becomes an Integer Linear Program (ILP). However, this makes solving it NP-complete. We therefore allow the cache size decision variables to take on any floating point value within the specified range. This relaxed LP problem can be solved in polynomial time. Subsequently, the calculated floating point cache sizes are rounded to the nearest integer value.

5.3.2 Identifying expanded path candidates

As a first step in the improvement process, the paths of the previous iteration are expanded with new storage sites. An end-to-end path Π_r of a request $r \in \boldsymbol{R}$ takes the form $\langle o_1, ..., o_n \rangle$,

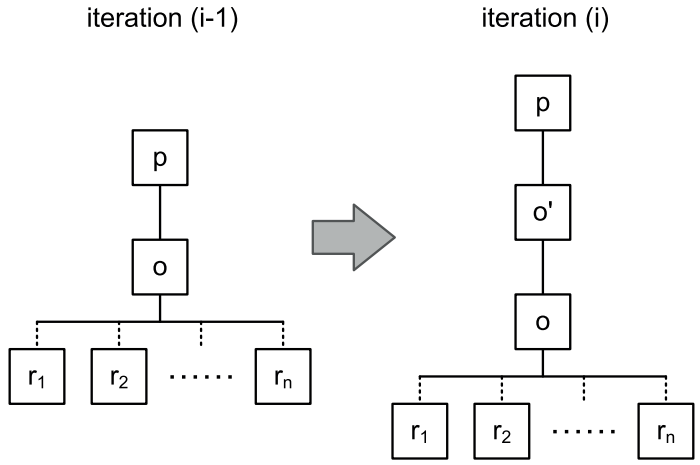


Fig. 5: A graphical example of how expanded path set candidates are created; The path set $\widehat{\Pi}_o$ is transformed into a new path set $\widehat{\Pi}_{o'}$; The dotted lines represent paths containing zero or more content caches

with o_1 the cache deployed in $p \in \boldsymbol{P}$ and o_n the cache deployed in a_r. At the start of the expansion process, all paths that share the same o_2 (i.e., the cache deployed in the successor domain of p) are grouped together. This means that all paths that were expanded with a shared cache in the previous iteration, will also be expanded with a shared cache in this iteration. We denote such a group of paths, with the top shared cache $o = o_2$, as $\widehat{\Pi}_o$ and call it a *sharing path set*. During iteration (i), the set of shared path sets $\{\widehat{\Pi}_o\}^{(i-1)}$ constructed during iteration $(i-1)$ is expanded with every $s \in \boldsymbol{S}$. An expanded shared path set is created by inserting a new shared cache o', deployed in s (i.e., $d_{o'} = s$), right after the root cache into all paths of the existing shared path set. The set $\{\widehat{\Pi}\}^{\text{cnd}}$ contains all the candidate shared path sets that are created during iteration (i). It is initialized to contain all path sets in $\{\widehat{\Pi}_o\}^{(i-1)}$. As described below, new candidate path sets are then iteratively added to it.

From here on, let us consider a single shared path set $\widehat{\Pi}_o \in \{\widehat{\Pi}_o\}^{(i-1)}$ and explain the expansion process for it. The process can subsequently be repeated for all other shared path sets. Concretely, the set is expanded with all $s \in \boldsymbol{S}$, that are not yet part of any path in the set. This means that an end-to-end path can never contain two different content caches deployed within the same storage site. The expansion of a path $\Pi = \langle o_1, o, ..., o_n \rangle \in \widehat{\Pi}_o$ with a content cache o', results in a path $\Pi' = \langle o_1, o', o, ..., o_n \rangle$. Performing this expansion for all paths in the set results in the new path set $\widehat{\Pi}_{o'}$. Subsequently, the cost-minimizing core Internet paths (cf., Section 5.1) and cache sizes (cf. Section 5.3.1) can be calculated for this new set. This, in turn, allows the total delivery cost of the set to be calculated (cf. Section 4.2). If the total delivery cost of $\widehat{\Pi}_{o'}$ is lower than that of $\widehat{\Pi}_o$, then this set becomes the optimal candidate set to replace $\widehat{\Pi}_o$. If the total delivery cost of $\widehat{\Pi}_{o'}$, minus the storage cost of o' is lower than that of $\widehat{\Pi}_o$, then the new set could potentially improve delivery costs if o' is shared across multiple shared path sets. If neither of these conditions is met, the new set cannot lower costs compared to iteration $(i-1)$ and it is discarded. In the former two cases, $\widehat{\Pi}_{o'}$ is added to the set $\{\widehat{\Pi}\}^{\text{cnd}}$ of candidate path sets for iteration (i).

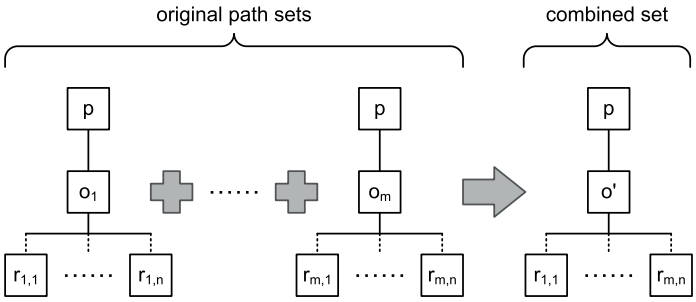


Fig. 6: A graphical example of how a group of path sets can be combined into a single path set with a shared cache; The original set of path sets $\{\widehat{\Pi}\}_s$ is transformed into a new combined path set $\widehat{\Pi}_{o'}$; The dotted lines represent paths containing zero or more content caches

Figure 5 further clarifies the creation of expanded path set candidates, using a graphical example. On the left is a shared path set $\widehat{\Pi}_o$, which contains a path for the requests $\{r_1,...,r_n\} \in \boldsymbol{R}$ and is represented as by tree. On the right is the same shared path set, after it has been extended with a new cache o'.

5.3.3 Identifying cache sharing opportunities

The end of the previous step results in a set of candidate expanded shared path sets $\{\widehat{\Pi}\}^{\text{cnd}}$. However, some of the candidate shared path sets could be combined into a single set, as their paths have the same storage site $s \in \boldsymbol{S}$ as a successor of $p \in \boldsymbol{P}$. The goal of this step is to identify all such compatible path sets, and create new path sets that combine them. Let us consider $\{\widehat{\Pi}\}_s \subseteq \{\widehat{\Pi}\}^{\text{cnd}}$ that contains all $\widehat{\Pi}_o \in \{\widehat{\Pi}\}^{\text{cnd}}$ for which $d_o = s$, with $s \in \boldsymbol{S}$ and $o \in \boldsymbol{O}$. As every path $\Pi \in \{\widehat{\Pi}\}_s$ shares s as the successor of p, they can be combined into a single shared path set. To achieve this, a new content cache $o' \in \boldsymbol{O}$, with $d_{o'} = s$, is constructed. Subsequently, every path $\Pi = \langle o_1, o, o_2, ..., o_n \rangle \in \{\widehat{\Pi}\}_s$, with $d_o = s$, is transformed into $\Pi' = \langle o_1, o', o_2, ..., o_n \rangle$ and added to the new shared path set $\widehat{\Pi}_{o'}$. Subsequently, $\widehat{\Pi}_{o'}$ is added to the set $\{\widehat{\Pi}\}^{\text{cnd}}$ of candidate path sets. This process can then be repeated for all $s \in \boldsymbol{S}$ in order to identify all combined shared path sets.

Figure 6 further clarifies the process of combining multiple shared path sets into a new combined path set with a shared cache. It shows a group of m shared path sets $\{\widehat{\Pi}\}_s = \{\widehat{\Pi}_{o_1}, ..., \widehat{\Pi}_{o_m}\}$, with $d_{o_1} = ... = d_{o_m} = s$. They are combined into a new shared path set $\widehat{\Pi}_{o'}$, where also $d_{o'} = s$.

The set $\{\widehat{\Pi}\}^{\text{cnd}}$ now contains all fully combined shared path sets, that combine *all* path sets with the same s as direct successor of p. However, the partially combined shared path sets that only contain a subset of the shared path sets with the same s can also be added to $\{\widehat{\Pi}\}^{\text{cnd}}$. This is done as follows. Consider the set $\{\widehat{\Pi}\}^{\text{cnd}}_r \subseteq \{\widehat{\Pi}\}^{\text{cnd}}$ that contains all the shared path sets $\widehat{\Pi}$ that contain an end-to-end path Π_r for request $r \in \boldsymbol{R}$. Such a set is either a direct expansion of a shared path set of iteration $(i-1)$ (cf. Section 5.3.2) or a combination of several such expanded sets (cf. Section 5.3.3). In the former case, no new paths are derived from it. In the latter case, the shared path set $\widehat{\Pi}$ consists of the union of several other, non-combined, shared path sets $\{\widehat{\Pi}_1, ..., \widehat{\Pi}_n\}$. A new set $\widehat{\Pi}'$, which combines

the same non-combined paths sets $\{\widehat{\Pi}_1,...,\widehat{\Pi}_n\}$ except for the one containing Π_r, is then constructed. Subsequently, $\widehat{\Pi}'$ is also added to $\{\widehat{\Pi}\}^{\text{cnd}}$.

The total number of candidate path sets in $\{\widehat{\Pi}\}^{\text{cnd}}$ can grow exponentially with the number of requests and storage sites. In order to reduce complexity of the problem, a subset of candidate paths can be filtered out. The algorithm uses a method that retains K path sets for every $r \in \boldsymbol{R}$. As the same path set can be retained for multiple requests, the total remaining number of path sets is therefore less than or equal to $|\boldsymbol{R}| \times K$. The K parameter thus allows the computational complexity of this step to be polynomially bound, offering a trade-off between execution time and optimality. For every request r, the K path sets that contain a path Π_r with the lowest delivery cost for r are retained.

5.3.4 Selecting optimal paths

The first two steps of the iterative improvement process generate a set of shared path sets $\{\widehat{\Pi}\}^{\text{cnd}}$. It potentially contains many path sets $\widehat{\Pi}$ that contains a path Π_r for request $r \in \boldsymbol{R}$. However, the final solution returned by iteration (i) should contain exactly one end-to-end path Π_r for every request r. The final step of the improvement process thus selects the optimal combination of path sets, such that exactly one path is selected for every request. This is an optimization problem, that can be formulated as an ILP formulation. The formulation consists of a boolean decision variable $p_{\widehat{\Pi}} \in \{0,1\}$ for every $\widehat{\Pi} \in \{\widehat{\Pi}\}^{\text{cnd}}$. This variable denotes whether or not the associated path set will be selected for the final solution. There is a single additional constraint, stipulating that for every request $r \in \boldsymbol{R}$ only one path set can be selected that contains a path Π_r for r:

$$\forall r \in \boldsymbol{R} : \sum_{\widehat{\Pi} \in \{\widehat{\Pi}\}^{\text{cnd}}_r} p_{\widehat{\Pi}} = 1 \tag{8}$$

The objective function is once again to minimize the total delivery cost of the solution. Let us define $\Delta(\widehat{\Pi})$, $\Psi(\widehat{\Pi})$ and $\Theta(\widehat{\Pi})$ as the total storage, processing and transmission costs of shared path set $\widehat{\Pi}$. The objective can then be formulated as follows:

$$\min \sum_{\widehat{\Pi} \in \{\widehat{\Pi}\}^{\text{cnd}}} p_{\widehat{\Pi}} \times \left(\Delta\left(\widehat{\Pi}\right) + \Psi\left(\widehat{\Pi}\right) + \Theta\left(\widehat{\Pi}\right) \right) \tag{9}$$

To calculate the value of the above objective function, the delivery cost of every $\widehat{\Pi} \in \{\widehat{\Pi}\}^{\text{cnd}}$ needs to be known. This is a costly operation, as for every $\widehat{\Pi}$ the LP formulation described in Section 5.3.1 needs to be solved. However, the cost of the non-combined expanded path sets (as calculated in Section 5.3.2) is known, as it needs to be calculated to determine whether or not the path set is a valid candidate. The number of paths in this set is much smaller and limited by $|\boldsymbol{S}| \times |\boldsymbol{R}|$. To avoid having to calculate the cost of all candidate shared path sets, we propose a method to estimate the total delivery cost of $\widehat{\Pi}$, based on the cost of the non-combined path sets of which it is composed. Every combined shared path set $\widehat{\Pi}_o$ equals the union of multiple non-combined shared path sets $\{\widehat{\Pi}_{o_1},...,\widehat{\Pi}_{o_n}\}$. The total delivery cost of $\widehat{\Pi}_o$ can then be estimated as follows:

$$\sum_{\widehat{\Pi}_{o_i} \in \{\widehat{\Pi}_{o_1},...,\widehat{\Pi}_{o_n}\}} \Delta\left(\widehat{\Pi}_{o_i}\right) + \Psi\left(\widehat{\Pi}_{o_i}\right) + \Theta\left(\widehat{\Pi}_{o_i}\right) - \frac{(n-1)}{n} \times \Delta\left(o_i\right) \tag{10}$$

With $\Delta\left(o_i\right)$ the storage cost associated with content cache $o_i \in \boldsymbol{O}$.

5.4 Input parameter estimation

In order to calculate the delivery costs, the presented algorithm expects several statistical input parameters, both related to the content provider and the customers. More specifically, the algorithm requires information about: average content duration, average simultaneously delivered content items and the cumulative popularity distribution. The average content duration can easily be derived from the content catalogue. On the other hand, the other two parameters are dynamic over time. The algorithm needs accurate dynamic predictions for these parameter, in order to be applicable in actual deployments. Note that the first step of the algorithm (cf. Section 5.2), which sets up direct core Internet paths between the content provider and access ISP, does not require this information to calculate the cheapest solution. Therefore, the content provider can first set up direct delivery paths. As time passes, it will be able to more accurately estimate values for these parameters, allowing it to iteratively improve the solution using the subsequent steps of the algorithm.

The modelling and prediction of popularity distribution curves of multimedia services has been an active research topic for several years [38,39]. Existing methods can thus be applied to estimate the cumulative popularity distribution. More recently, some methods have been proposed to predict the popularity of individual content items [40,41]. This can be applied to predict the number of simultaneous requests originating from an access ISP. Alternatively, information from the recent past can be used as a fast estimation of these parameters in the near future.

5.5 Computational complexity

In this section, the worst-case time complexity of the different steps of the algorithm is presented. As they are performed in sequence, the worst case complexity equals that of the most expensive step. Finding QoS-constrained paths is part of several of the algorithm's steps. Therefore, we first analyse the time complexity of solving the MCOP problem. A wide range of optimal, as well as heuristic algorithms have been presented to solve this problem. As recently surveyed by Garroppo *et al.* [42], modern near-optimal approximation heuristics can achieve polynomial worst-case time complexity in terms of number of nodes and links, and exponentially in terms of the number of constraints. The aim of the algorithm is to find a path of transit ISPs between the content provider p_r and access ISP a_r associated with request r. As such, the number of nodes equals the total number of transit ISPs $|\boldsymbol{T}|$. If we assume the set of links between transit ISPs is defined as $\boldsymbol{L}$, then the number of links is defined as $|\boldsymbol{L}|$. Finally, the number of constraints equals the number of QoS parameters $|\boldsymbol{Q}|$. Consequently, based on the reported time complexities [42], the worst-case time complexity of executing the MCOP algorithm becomes:

$$O\left(|\boldsymbol{T}| \times |\boldsymbol{L}| \times (|\boldsymbol{T}| \times \log|\boldsymbol{T}|)^{|\boldsymbol{Q}|-1}\right) \tag{11}$$

The first step of the algorithm, that sets up delivery paths without intermediary caches, simply executes the MCOP heuristic for each request $r \in \boldsymbol{R}$. In combination with Eq. 11, this gives a worst-case time complexity defined as follows:

$$O\left(|\boldsymbol{R}| \times |\boldsymbol{T}| \times |\boldsymbol{L}| \times (|\boldsymbol{T}| \times \log|\boldsymbol{T}|)^{|\boldsymbol{Q}|-1}\right) \tag{12}$$

The subsequent iterative improvement process consists of another three steps. During several of these steps, the optimal cache sizes of the storage sites along the delivery paths

need to be calculated. This is done by solving a relaxed LP, which can be solved in polynomial time using Karmarkar's algorithm [43], as a function of the number of decision variables. The cache size problem has one decision variable for each of the caches (i.e., storage sites and access ISPs) along the end-to-end delivery path. If we assume that setting up the initial delivery paths is iteration 0, then the maximum number of caches on any single path during iteration i of the algorithm equals $i+1$. As such, the worst-case time complexity of calculating the optimal cache sizes can be defined a function of the current iteration i as follows:

$$O\left((i+1)^{3.5}\right) \tag{13}$$

In the first step of the iterative improvement process, the delivery paths calculated during the previous iteration are extended with all feasible storage sites. There is one path for each request in $\boldsymbol{R}$, which is extended for at most every storage site in $\boldsymbol{S}$. For each of these extended path candidates, the core Internet path is calculated for each link in the end-to-end path. As the number of storage sites along this path is directly proportional to the current iteration i, so are the number of links in the end-to-end path. For each end-to-end path, the maximum core Internet paths that need to be calculated using the MCOP algorithm equals $i+1$. Moreover, for each of these end-to-end paths, the optimal cache sizes need to be calculated. Taking this into account, in combination with Eq. 11 and Eq. 13, this step's worst-case time complexity equals:

$$O\left(\boldsymbol{R}\times\boldsymbol{S}\times\left((i+1)\times|\boldsymbol{T}|\times|\boldsymbol{L}|\times(|\boldsymbol{T}|\times\log|\boldsymbol{T}|)^{|\boldsymbol{Q}|-1}+(i+1)^{3.5}\right)\right) \tag{14}$$

Second, cache sharing opportunities are identified. In this step, the extended paths from the first iterative step are combined when they share a common storage site. The highest number of combinations occurs if they can be combined in pairs. As the maximum number of combinable paths equals $|\boldsymbol{R}|\times|\boldsymbol{S}|$, this results in a number of combinations equal to:

$$\frac{|\boldsymbol{R}|\times|\boldsymbol{S}|}{2} \tag{15}$$

Additionally, for each of these combinations several sub-combinations can be created, by removing one of the request paths, which results in a number of additional combinations:

$$\frac{|\boldsymbol{R}|^2\times|\boldsymbol{S}|}{2}$$

Note that this is a broad upper bound, as the above procedure results in invalid combinations as well as duplicates, which are both filtered out. The calculations performed for all end-to-end paths in the first iterative improvement step (i.e., calculating core paths and optimal cache sizes) do not need to be repeated here, due to the cost estimation method introduced in Section 5.3.4. This results in the following worst-case time complexity:

$$O\left(\frac{(|\boldsymbol{R}|+1)\times|\boldsymbol{R}|\times|\boldsymbol{S}|}{2}\right) \tag{16}$$

Third, the optimal path for each request is selected. As the total number of path candidates resulting from the previous steps can become very high, only the K most promising paths for each request in $\boldsymbol{R}$ are selected. This results in a total of $K\times\boldsymbol{R}$ remaining paths. The algorithm solves this problem by formulating it as an ILP, which has a worst-case exponential time complexity, as a function of the number of decision variables [44]. As there is one

decision variable for each candidate path set, this results in the following worst-case time complexity:

$$O\left(2^{K\times \boldsymbol{R}}\right) \tag{17}$$

In summary, we have shown that setting up the initial delivery paths scales polynomially in terms of the size of the network as well as the number of requests (cf. Eq. 12), while it scales exponentially in terms of the number of QoS parameters. However, as the number of QoS parameters is usually bound by a relatively low value, this is expected to be of limited influence. The first iterative improvement step has a higher computational complexity, but also scales polynomially in terms of the network size and request count, and exponentially in terms of number of QoS parameters (cf. Eq. 14). Additionally, its theoretical worst-case time complexity increases with the number of performed iterations. The second improvement step has a lower computational complexity, depending polynomially only on the request count and number of storage sites (cf. Eq. 16). Finally, the third improvement step requires an ILP formulation to be solved. Solving it optimally scales exponentially in terms of the number of requests (cf. Eq. 17). However, several heuristics exist for finding feasible sub-optimal solutions for ILP models [45,46], which could be employed to reduce the algorithm's worst-case time complexity to polynomial in terms of the request count at the cost of optimality.

6 Results & Discussion

This section evaluates the heuristic presented in Section 5. First, the scalability is evaluated. Second, the merits of our approach are validated under a variety of conditions. More specifically, the usefulness of dynamically deployed intermediary caches and cache sharing is evaluated, as these are the novel aspects of our approach compared to existing end-to-end content delivery mechanisms. Additionally, we determine the impact on caching efficiency of several parameters, such as the location of access ISPs and storage sites and the storage cost. The presented results were obtained from a Java-based implementation of the algorithm. The LP optimization problems were solved using the Java version of CPLEX 12.3[1]. All tests were performed on a computer with one Dual-Core AMD Opteron 2212 processor and 4 GiB RAM memory, running the GNU/Linux Debian 6.0 operating system. All depicted results are averaged over 100 iterations, with the error bars showing the standard error of the mean. The algorithm was run for two rounds. First, the initial delivery paths were calculated. Second, a single iterative improvement step was performed. This means that the final end-to-end delivery paths consist of at most one intermediary storage site. The parameter K (cf. Section 5.3.3) is set to 100 throughout the evaluation.

6.1 Evaluation scenario

The core Internet topology used throughout the evaluations, was generated using the ReaSE topology generator [47]. It consists of 250 ASes, including 45 transit domains and 205 stub domains. The ReaSE parameters P and Δ were left at the default values 0.4 and 0.04. The total diameter of the generated core network is 4 hops. Every generated AS corresponds to a single transit ISP $t \in \boldsymbol{T}$. As the algorithm calculates delivery paths for every content provider separately, we consider only a single content provider $p \in \boldsymbol{P}$ without loss of generality. The transit ISP gateway t_p of p is randomly selected from the 205 stub transit ISPs. Subsequently,

[1] `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`

the set of access ISPs $\boldsymbol{A}$ is created. The number of access ISPs is set to 10, unless stated otherwise. One of the goals of this evaluation is to assess the effect of the position of access ISPs within the network on the merits of cache sharing. As such, we define an *access ISP vicinity* parameter $av \in [0,1]$. This parameter defines the probability that access ISPs are positioned near each other in the network. Concretely, the gateway of every access ISP is determined as follows. First, a random gateway t_a is selected for the first access ISP $a \in \boldsymbol{A}$ from the set of stub transit ISPs. The gateway for all other access ISPs $a_i \in \boldsymbol{A}$ is selected as follows. With a probability $av_1 = av$, a random stub transit ISP that is exactly one hop away from t_a is selected as the gateway of a_i. The probability that a stub transit ISP n hops away is selected, is calculated as follows:

$$av_n = \left(1 - \sum_{i=1}^{n-1} av_i\right) \times av \tag{18}$$

If the maximum hop distance from t_a equals m, then $av_m = (1 - \sum_{i=1}^{m-1} av_i)$ to make sure a gateway is definitely selected. As an example, if $av = 0.7$ and $m = 4$, then the access ISP vicinity probabilities become 0.7, 0.21, 0.063 and 0.027. Throughout the rest of this section $av = 0.7$ unless otherwise stated. Finally, the storage sites $\boldsymbol{S}$ are added to the network. Unless stated differently, 10 storage sites are used. We now define the parameter $sv \in [0,1]$ as the *storage site vicinity*. The gateway t_s for every storage site $s \in \boldsymbol{S}$ is selected as follows. First, an access ISP $a \in \boldsymbol{A}$ is selected in round robin fashion. With a probability of $sv_0 = sv$ the gateway t_a of a is selected as the gateway t_s of s. The probability that a gateway n hops away from t_a is selected is then calculated as follows:

$$sv_n = \left(1 - \sum_{i=0}^{n-1} sv_i\right) \times sv \tag{19}$$

Again, if the maximum hop distance from t_a equals m, then $sv_m = (1 - \sum_{i=0}^{m-1} sv_i)$. If the value of sv is chosen close to 1, then most access ISPs will have a storage site nearby in the network. When the value of sv is close to 0, storage sites will be further away. Unless stated differently, $sv = 0.7$ throughout the rest of the evaluation.

The evaluation scenario considers a VoD service, with the content provider's catalogue consisting of 5000 unique movies. The average movie duration is 5400 seconds, or 90 minutes. As the algorithm calculates a separate solution for every bit-rate, we can consider a single bit-rate without loss of generality. The movie bit-rate is set to 5 Mbps. In literature, several models have been presented for representing the cumulative popularity distribution of multimedia services. We use the Zipf-mandelbrot distribution, with $\alpha = 0.8$ and $q = 1$ [48].

Every transit ISP is characterised by a set of QoS classes $\boldsymbol{C}$. As stated, the presented model supports an arbitrary number of QoS parameters. However, as a TCP-based progressive download scenario is assumed, we do not need to consider packet loss and focus the *delay* and *availability* QoS parameters in this evaluation. The transit ISPs support three delay values: 0.001, 0.005 and 0.01 seconds. Additionally, two availability values are supported: 0.999 and 0.9999. They respectively result in an average yearly downtime of 8.76 hours and 52.56 minutes. The reservation cost of a QoS class $c \in \boldsymbol{C}$ is calculated as follows:

$$\theta_c = \frac{0.00001}{\text{delay}_c \times (1 - \text{availability}_c)} \tag{20}$$

This reservation cost is paid for every MBps (megabyte per second) that is transferred through the associated ISP domain. In total, every transit ISP offers all 6 combinations of

Table 2: The QoS classes used in the evaluation scenario

delay	availability	θ
0.01	0.999	1
0.005	0.999	2
0.01	0.9999	10
0.001	0.999	10
0.005	0.9999	20
0.001	0.9999	100

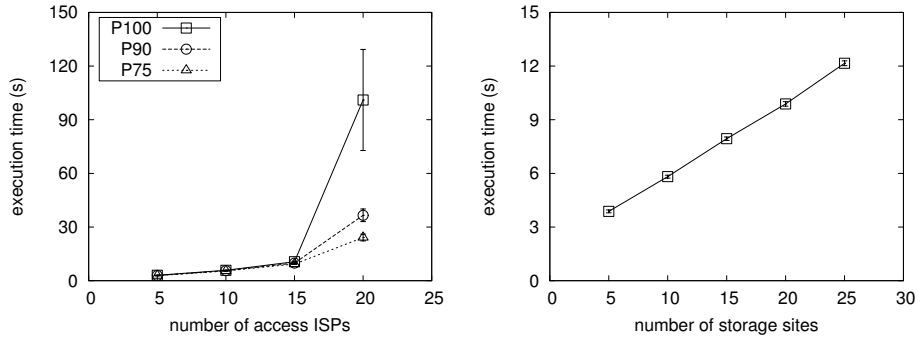
these QoS parameter values, as depicted in Table 2. The cheapest and most expensive QoS classes thus result in a cost of 0.625 and 62.5 per ISP domain per content stream per second. The storage cost λ_s is the same for all storage sites $s \in \boldsymbol{S}$. As the effect of this cost is studied in this section, it is a variable parameter *sc*. Its default value is 0.0001, which gives a storage cost per movie of 0.3375 per second. To be able to more accurately study the synergy between the storage and transmission costs, the processing cost is assumed to be negligible throughout this evaluation (i.e., $\varphi_s = 0$ for all $s \in \boldsymbol{S}$). Caches can also be deployed within the access ISP domain, without an additional storage cost. Such caches are only considered when specifically stated. Otherwise, it is assumed $\sigma_a = 0$ for all $a \in \boldsymbol{A}$.

Finally, the scenario contains a set of requests $\boldsymbol{R}$ for setting up QoS-constrained end-to-end delivery paths. A single request $r \in \boldsymbol{R}$ is generated for every access ISP $a \in \boldsymbol{A}$. For every request, the average number of simultaneous content streams is determined uniformly at random from the range $[50, 250]$. The requested end-to-end delay is chosen uniformly at random from the range $[0.005, 0.05]$ seconds, while the availability is selected from $[0.995, 0.9995]$. The least stringent constrains, with delay 0.05 seconds and availability 0.995, can be satisfied on a path of up to 5 transit ISPs long with the cheapest QoS class. On the other hand, on a path of up to 5 transit ISPs, the most stringent constraints (i.e., delay $= 0.005$ and availability $= 0.9995$) can only be achieved when using the most expensive QoS class in all domains along the path.

6.2 Scalability

In this section, the algorithm's computational performance and scalability are evaluated, in terms of execution time. Its computational complexity is influenced by the number of access ISPs and storage sites within the network. Figure 7 depicts the execution time as a function of the number of access ISPs and storage sites. The depicted execution time is for the first iterative improvement round of the algorithm. The execution time of the initial set up process was always less than 1 second and is therefore not depicted. This allows the algorithm to be applied to highly dynamic situations, as the initial solution can be swiftly calculated to set up the initial direct core Internet paths between the content provider and access ISPs. Over time the slower improvement steps of the algorithm can then be employed to iteratively add storage sites.

Increasing the number of access ISPs complicates the end-to-end content delivery problem in several ways. First, the number of decision variables in the LP problem to determine the optimal cache sizes increases linearly. Additionally, the number of such problems that need to be solved increases linearly as well. As pure LP formulations can be solved in polynomial time [43], this results in a (non-linear) polynomial increase in execution time. The number of core Internet paths that need to be calculated also increases linearly. The complexity increase of this step depends on the complexity of the MCOP algorithm. However,



(a) as a function of access ISP count (b) as a function of storage site count

Fig. 7: Execution time of the algorithm

as many polynomial time heuristics exist to solve the MCOP problem, this also results in a polynomial increase in execution time. On the other hand, the number of expanded path candidates that exist increases exponentially. Additionally, as the LP problem to determine the set of cost minimizing expanded paths contains integer variables, solving it takes exponential time in the worst case. The exponential increase in the amount of candidate paths can be countered by using the parameter K to limit the subset of considered candidate expanded paths (cf. Section 5.3.3). This reduces the increase to a linear one, and in turn limits the complexity of the ILP problem to select the optimal candidates. Nevertheless, the worst case time complexity of the ILP formulation remains exponential. Figure 7a depicts the execution time (in seconds) as a function of the number of access ISPs. As expected from the analysis above, the scaling behaviour is worse than linear. At 20 access ISPs the execution time explodes. However, the standard deviation also grows significantly, suggesting a high variability in the results. Closer inspection shows that this is indeed the case. The execution time of 65 of the 100 runs was below 50 seconds, while only 16 runs showed an execution time above 100 seconds of which 4 were above 300 seconds. The two slowest runs took 1583 and 1845 seconds respectively. To further illustrate this, the graph also shows the $P90$ and $P75$ curves, which are averages over the 90 and 75% best values respectively. These two curves depict a much lower standard error and significantly reduced average execution time for 20 access ISPs (i.e., 36 and 24 seconds respectively). This high variability shows that the ILP formulation can often be solved within a feasible time frame. However, on some rare occasions it takes exponential time to find the optimal solution. This can be prevented by configuring the ILP solver with a maximum calculation time. This will solve the execution time variability, but will in rare cases cause the solver to only find a suboptimal solution.

The number of storage sites also affects the problem complexity. It also causes the number of candidate expanded trees to grow exponentially. However, by selecting the subset of K most promising candidates, the number of candidates considered in the ILP formulation remains constant. Additionally, in contrast to the number of access ISPs, the complexity of the LP formulation is not increased. Instead only the number of times it needs to be solved grows. This is also reflected in the results in Figure 7b, which depicts the execution time (in seconds) as a function of the number of storage sites. In line with the above analysis, the algorithm scales linearly with the number of storage sites.

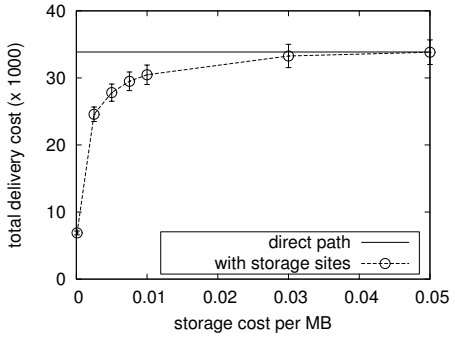


Fig. 8: Influence of the storage cost *sc* on the merits of intermediary content caches; comparing end-to-end QoS-aware content delivery with and without intermediary storage sites

6.3 Storage site merits

An important novel aspect of the presented approach is the inclusion of cloud-based storage sites within the end-to-end federations. We intuitively expect this to reduce the transmission costs associated with the delivery of multimedia content. Nevertheless, the conditions under which this is the case are unclear. This section assesses the effect of several input parameters on the merits of intermediary storage sites. More specifically, it is expected both the storage cost *sc* and storage site vicinity *sv* will affect the usefulness of storage sites. Figure 8 compares the delivery cost of the solution with and without intermediary storage sites as a function of the storage cost *sc*. Subsequently, Figure 9 shows how the vicinity of storage sites to the access ISPs (i.e., *sv*) affects the total delivery cost and caching efficiency.

Intuitively, it is expected intermediary storage sites are only useful if the storage cost is below a certain threshold relative to the transmission cost. In order to determine this threshold, Figure 8 compares the total delivery cost to all access ISPs with and without intermediary content caches. As end-to-end QoS negotiation mechanisms traditionally reserve a path through the Internet core directly from the content provider to its customers, the solution without storage sites is comparable to those traditional methods. The delivery cost of the solution without intermediary storage sites is independent of the storage cost. This results in a constant total cost value of 33860 in the evaluated scenario. As our algorithm starts from this solution as well and then iteratively improves it, it can never perform worse. Additionally, the graph shows that it performs significantly better when the storage cost is less than 0.03 per MB (megabyte), or on average 101.25 per movie. Additionally, it can be calculated that the average transmission cost without an intermediary cache is on average about 20 per movie, in the evaluated scenarios. These results therefore show that deploying intermediary content caches can significantly reduce delivery costs, even when the average cost for storing a content items is several times higher than the cost for transmitting one. If the storage cost is 0.01 per MB, the total cost is about 10% better than that of the solution without storage sites. When the storage cost becomes very small (i.e., 0.0001 per stored MB or 0.3375 per movie), the total delivery cost can be reduced down to on average 6883 in the evaluated scenarios. This is a reduction of 80% compared to the traditional QoS negotiation approaches that only employ direct end-to-end paths.

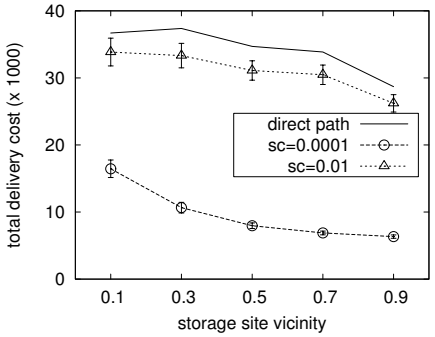


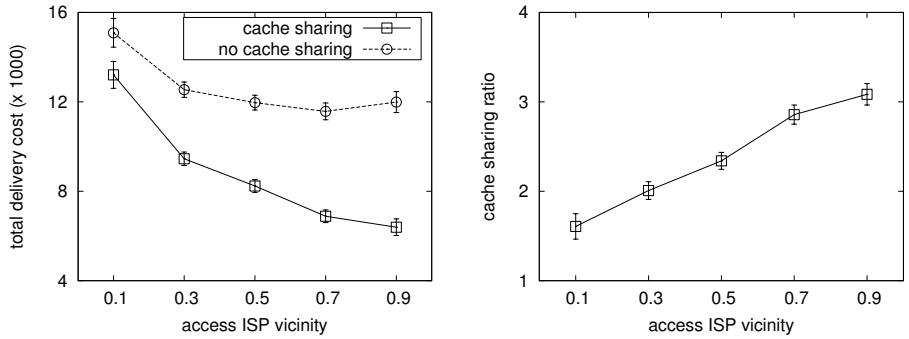
Fig. 9: Influence of the storage site vicinity *sv* on the merits of intermediary content caches; comparing end-to-end QoS-aware content delivery with and without intermediary storage sites

If storage sites are located closer to the access ISPs, content will on average need to traverse a shorter path through the Internet core. This increases the effectiveness of intermediary content caches and is thus expected to reduce transmission costs. Figure 9 compares the total delivery cost of the solution with and without including intermediary content caches. As expected, positioning the storage sites closer to the access ISPs (i.e., increasing the value of *sv*) significantly reduces the total delivery costs. If storage providers are positioned far away from the access ISPs and the storage cost is not very low (i.e., $sc = 0.01$), then using storage sites cannot reduce delivery costs as compared to using the direct path from content provider to access ISP. However, if $sv = 0.5$, which gives an 87.5% chance that a storage site is within three hops of every access ISP, our algorithm significantly outperforms the traditional direct end-to-end path even if the storage cost is not very low. If the storage cost is very low (i.e., $sc = 0.0001$), the traditional approach is outperformed independent of the value of *sv*.

In summary, it can be concluded that the use of storage sites for deploying dynamic caches inside the Internet core can indeed outperform the traditional QoS-aware end-to-end delivery approach that directly sends content from the provider to its customers. Nevertheless, the significance of the achieved cost reduction depends on several factors. If the cost for storing a content item becomes very high relative to the cost for transmitting it, caching no longer reduces costs. Additionally, unless the storage cost is negligible compared to the transmission cost, storage sites need to be positioned relatively close to the access ISPs. On the other hand, if the storage cost is insignificant relative to the transmission cost, randomly placed storage sites remain useful in reducing the total delivery cost.

6.4 Cache sharing merits

A major advantage of deploying content caches in intermediary domains across the Internet, is the opportunity to share them among several access ISPs. This greatly improves the cache's efficiency, as more content can be served from it without increasing the total storage cost. Nevertheless, the gain that can be achieved through cache sharing is influenced by factors such as the access ISP vicinity *av*. This section evaluates the effect of *av* on cache sharing in more detail. As metrics to evaluate cache sharing, the total delivery cost and cache



(a) effect on total delivery cost

(b) effect on total cache reuse

Fig. 10: Influence of the access ISP vicinity av on the efficiency of cache sharing

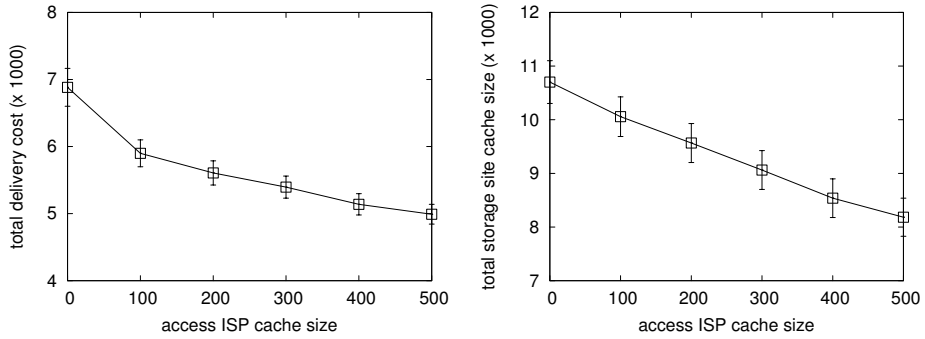
sharing ratio are used. The cache sharing ratio is defined as the average number of access ISPs that share a storage site content cache. It thus takes a value between 1 (i.e., no cache sharing) and $|\boldsymbol{A}|$ (i.e., the cache is shared by all access ISPs). Figure 10 shows the total delivery cost and cache sharing ratio as a function of access ISP vicinity av.

Figure 10a compares the total delivery cost, with and without cache sharing, as a function of the access ISP vicinity av. Previously, it was shown that content caches can be more efficiently employed if they are positioned near the access ISPs (cf. Section 6.3). As a consequence, we intuitively expect cache sharing to be more efficient if the access ISPs are positioned close together. This is reflected in the results shown in the figure. If the access ISPs are far away from one another (i.e., $av = 0.1$), the solution with cache sharing has only a slightly better delivery cost than the solution without cache sharing (i.e., a 12.5% reduction). However, as the access ISPs are positioned closer together, more cache sharing opportunities become available. If $av = 0.9$, then the total delivery cost of the solution with cache sharing is 46% reduced compared to the solution without cache sharing. This gain is entirely caused by the reduction in storage costs cache sharing introduces. The cache sharing ratio is depicted in Figure 10b. This figure confirms our previous findings, showing that more cache sharing occurs as the access ISPs are positioned closer together.

In summary, the presented results prove that deploying content caches in intermediary domains, as opposed to at the client side, does have its advantages. Specifically, it allows those caches to be shared among different access ISPs, reducing the storage costs. Results show that cache sharing is most effective when several access ISPs are positioned close together in the network. For access ISPs further away from each other, cache sharing is not useful, as the shared content cache will be too far away from at least part of the shared access ISPs.

6.5 Caching in the access network

In addition to the dynamic content caches deployed across the Internet, the access ISP can also deploy a local cache. This locally cached content does not need to traverse any intermediary transit ISP domains, further reducing the total transmission cost compared to intermediary deployed caches. On the other hand, these locally deployed caches cannot be



(a) effect on total delivery cost (b) effect on storage site cache usage

Fig. 11: Influence of caching in the access domain on caching efficiency of the storage sites

shared, potentially reducing their efficiency. Deploying caches within the access domain is known to be expensive [49]. We thus expect the size of such a cache to be relatively small. Figure 11b explores the effect of such a small access ISP cache on the total delivery cost and the total size of the storage site caches.

Intuitively, we expect that caching in the access ISP network will reduce the total transmission costs. Additionally, as the intermediary caches will become smaller, the storage cost is also expected to drop. Figure 11a depicts the total delivery cost as a function of an increasing access ISP cache size. Note that such a cache of the same size is deployed in every access domain. As shown in the figure, deploying access caches significantly reduces the total delivery cost. A cache of 500 items in every access domain results in a cost reduction of 27%. However, this is only assuming that no cost is associated with the access caches. As the scenario consists of 10 access ISPs, they have a combined cache size of 5000 items. In this specific scenario, the storage cost is 0.0001. If these access caches had the same cost as the intermediary storage site caches, this would result in a total storage cost of 1687.5. If this is added to the total delivery cost, then the total achieved cost reduction is reduced to a mere 3%, as opposed to 27%. Figure 11b presents the total size of the content caches deployed in the intermediary storage sites. Deploying access caches of 500 items each, reduces the total in-network cache size from 10702 to 8183. The sum of the in-network and access cache sizes is thus higher than the total cache size when only using in-network caches.

In summary, we can conclude that small static caches deployed in the access domain can reduce the total delivery costs. However, this reduction is only significant if there is no cost associated with the caches deployed in the access network.

7 Potential Extensions

The algorithms presented in Section 5 make some assumptions concerning the way content is encoded and delivered. In this section, we show how these algorithms can be trivially extended to support scalable, as well as segmented video, and thus be more generally applicable.

7.1 Scalable video coding

It was assumed that the different bit-rate versions of the same multimedia content item are unrelated. As such, the calculation of end-to-end delivery paths can be done separately for each bit-rate, without taking other bit-rates into account. This is a valid assumption for traditionally encoded content. Recently, Scalable Video Coding (SVC) has garnered a lot of attention [50]. It allows video content to be encoded as a set of layers, that can be combined to increase quality. The base layer can thus be decoded separately, resulting in a low quality video. If, however, it is decoded together with one or more enhancement layers, video quality increases. Obviously, when content is encoded using SVC the assumption that different bit-rate versions of the same content item are unrelated no longer holds.

Our presented model and algorithm can be easily adapted to support SVC encoded content as follows. When an access ISP requests a new QoS-aware end-to-end delivery path, it no longer creates a single request $r \in \boldsymbol{R}$. Instead, it creates a set of requests, one for each layer of the SVC encoded content it wants to receive. For example, assume a content provider offers its content catalogue encoded using SVC in three layers with bit-rates b_1, b_2 and b_3. If an access ISP wants to set up a delivery path for the lowest quality version, it sends the content provider a request r with bit-rate $b_r = b_1$. If however it wants to request the highest quality, it sends three requests, r_1, r_2 and r_3. Their bit-rates are respectively $b_{r_1} = b_1$, $b_{r_2} = b_2$ and $b_{r_2} = b_2$. The advantage of SVC-based content is that different quality requests now partially overlap, resulting in more cache sharing opportunities and thus decreased delivery costs.

7.2 Segment-based content delivery

The presented model and algorithm assumes the delivered content items to be monolithic units of data. However, it has been shown in practice that multimedia content often has a high internal popularity skew [51]. For example, in many video-based multimedia services, the beginning of videos is often much more popular than the end. As such, it has been shown that splitting content into temporal segments can significantly increase delivery and caching performance [52]. Although the presented model is based on content as the unit of data, it can be adapted to support temporal content segments. The only required change is in the statistical information provided by the content provider. Specifically, the weighted average content duration should be calculated on a per-segment basis, instead of per-content item, the number of items in the catalogue should be replaced with the total number of segments, and the popularity distribution function should model the segment popularity instead of entire content item popularity.

8 Conclusion

This article presents a novel framework for setting up end-to-end federations between the stakeholders involved in the delivery of multimedia content. More specifically, it guides the negotiation of SLAs between content providers, ISPs and cloud-based storage sites. This allows them to overcome the disadvantages associated with current delivery approaches, such as OTT content provisioning and content offered directly by access ISPs over a managed IP network. In contrast to existing works, our framework includes storage sites in the end-to-end delivery paths, allowing content caches to be dynamically deployed throughout the

network. This introduces an additional complexity to the problem, but allows further optimization of the delivery process. This article proposes a detailed mathematical model to optimize the content provider's end-to-end delivery costs. An optimization algorithm is presented for solving the model. It satisfies the customer's requested QoS, while minimizing the total delivery cost. To achieve this, it determines optimal QoS-constrained routes through the Internet core, and identifies well positioned intermediary storage sites for the deployment of content caches. Additionally, the algorithm calculates the amount of resources that need to be reserved along these routes and within the identified storage sites.

The presented framework was thoroughly validated based on evaluation results. The algorithm's scalability was characterized and the merits of our novel approach, that includes intermediary content caches, were quantified. Results show that including intermediary storage sites within the end-to-end delivery paths can significantly reduce delivery costs compared to traditional end-to-end QoS reservation mechanisms that use only direct QoS-constrained paths between the content provider and its customers. The significance of the cost reduction does depend on some external factors, such as the cost for storing an item, as compared to transmitting it and the vicinity in the network of storage sites to the access ISPs. Even if the storage cost per content item is higher than the transmission cost per item, a cost reduction of 10% can be easily achieved. If the storage cost becomes a fraction of the transmission cost, this reduction reaches up to 80% in the evaluated scenario. Additionally, as these content caches are deployed inside the network, they can be shared among different access ISP customers. The results prove that cache sharing significantly decreases the delivery costs for access ISPs that are located near each other. In the evaluated scenario, cache sharing resulted in an additional cost reduction of up to 46%.

Acknowledgements Jeroen Famaey is partially funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT). Steven Latré and Tim Wauters are partially funded by the Fund for Scientific Research Flanders (FWO). The research leading to these results was partially performed within the context of the FP7 OCEAN project and received funding from the European Union's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement number 248775.

References

1. Amigo, I., Belzarena, P., Larroca, F., Vaton, S.: Network bandwidth allocation with end-to-end QoS constraints and revenue sharing in multi-domain federations. In: Proceedings of the 7th International Conference on Internet Charging and QoS Technologies, pp. 50–62 (2011). DOI 10.1007/978-3-642-24547-3_6
2. Serrano, M., van der Meer, S., Holum, V., Murphy, J., Strassner, J.: Federation, a matter of autonomic management in the Future Internet. In: Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 845–849 (2010). DOI 10.1109/NOMS.2010.5488357
3. Kumar, N., Saraph, G.: End-to-end QoS in interdomain routing. In: Proceedings of the 2nd International Conference on Networking and Services (2006). DOI 10.1109/ICNS.2006.45
4. Xiangjiang, H., Peidong, Z., Kaiyu, C., Zhenghu, G.: AS alliance in inter-domain routing. In: Proceedings of the 22nd International Conference on Information Networking and Applications – Workshops (AINAW), pp. 151–156 (2008). DOI 10.1109/WAINA.2008.209
5. Roberts, L.: A radical new router. IEEE Spectrum **46**(7), 34–39 (2009). DOI 10.1109/MSPEC.2009.5109450
6. Pouyllau, H., Carofiglio, G.: Inter-carrier SLA negotiation using Q-learning. Telecommunication Systems (2011). DOI 10.1007/s11235-011-9505-5
7. Brooks, P., Hestness, B.: User measures of quality of experience: Why being objective and quantitative is important. IEEE Network **24**(2), 8–13 (2010). DOI 10.1109/MNET.2010.5430138
8. Famaey, J., Latré, S., Wauters, T., De Turck, F.: FedRR – a federated resource reservation algorithm for multimedia services. In: Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS) (2012)

9. Famaey, J., Latré, S., Wauters, T., De Turck, F.: An SLA-driven framework for dynamic multimedia content delivery federations. In: Proceedings of the Fifth International Workshop on Distributed Autonomous Network Management Systems (DANMS) (2012)
10. Pouyllau, H., Douville, R.: End-to-end QoS negotiation in network federations. In: Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 173–176 (2010). DOI 10.1109/NOMSW.2010.5486578
11. Korkmaz, T., Krunz, M.: Multi-constrained optimal path selection. In: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pp. 834–843 (2001). DOI 10.1109/INFCOM.2001.916274
12. Chen, S., Nahrstedt, K.: On finding multi-constrained paths. In: Proceedings of the IEEE International Conference on Communications, pp. 874–879 (1998). DOI 10.1109/ICC.1998.685137
13. De Neve, H., Van Mieghem, P.: A multiple quality of service routing algorithm for PNNI. In: Proceedings of the IEEE ATM Workshop, pp. 324–328 (1998)
14. Liu, G., Ramakrishnan, K.: A*Prune: an algorithm for finding K shortest paths subject to multiple constraints. In: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pp. 743–749 (2001). DOI 10.1109/INFCOM.2001.916263
15. Yuan, X., Liu, X.: Heuristic algorithms for multi-constrained quality of service routing. In: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pp. 844–853 (2001). DOI 10.1109/INFCOM.2001.916275
16. Xiao, J., Boutaba, R.: QoS-aware service composition and adaptation in autonomic communication. IEEE Journal on Selected Areas in Communications **23**(12), 2344–2360 (2005). DOI 10.1109/JSAC.2005.857212
17. Yan, J., Kowalczyk, R., Lin, J., Chhetri, M.B., Goh, S.K., Zhang, J.: Autonomous service level agreement negotiation for service composition provision. Future Generation Computer Systems **23**, 748–759 (2007). DOI 10.1016/j.future.2007.02.004
18. Balasubramaniam, S., Botvich, D., Carroll, R., Mineraud, J., Nakano, T., Suda, T., Donnelly, W.: Biologically inspired future service environment. Computer Networks **55**(15), 3423–3440 (2011). DOI 10.1016/j.comnet.2011.07.004
19. Yuanming, C., Wendong, W., Xiangyang, G., Xirong, Q.: Initiator-domain-based SLA negotiation for inter-domain QoS-service provisioning. In: Proceedings of the 4th International Conference on Networking and Services (2008). DOI 10.1109/ICNS.2008.43
20. Rubach, P., Sobolewski, M.: Dynamic SLA negotiation in autonomic federated environments. In: Proceedings of On the Move to Meaningful Internet Systems (2009). DOI 10.1007/978-3-642-05290-3_36
21. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification (WS-Agreement) (2011). `http://www.ogf.org/documents/GFD.193.pdf`
22. Battré, D., Brazier, F., Clark, K., Oey, M., Papaspyrou, A., Wieder, P., Ziegler, W.: WS-Agreement negotiation version 1.0 (2011). `http://www.ogf.org/documents/GFD.192.pdf`
23. Hudert, S., Ludwig, H., Wirtz, G.: Negotiating SLAs – an approach for a generic negotiation framework for WS-Agreement. Journal of Grid Computing **7**(2), 225–246 (2009). DOI 10.1007/s10723-009-9118-3
24. Hasselmeyer, P., Mersch, H., Koller, B., Quyen, H.N., Schubert, L., Wieder, P.: Implementing an SLA negotiation framework. In: Proceedings of Expanding the Knowledge Economy: Issues, Applications, Case Studies (eChallenges), pp. 154–161 (2007)
25. Parkin, M., Hasselmeyer, P., Koller, B., Wieder, P.: An SLA re-negotiation protocol. In: Proceedings of the 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (2008)
26. Passarella, A.: A survey on content-centric technologies for the current internet: CDN and P2P solutions. Computer Communications **35**(1), 1–32 (2012). DOI 10.1016/j.comcom.2011.10.005
27. Mao, Z.M., Cranor, C.D., Douglis, F., Rabinovich, M., Spatscheck, O., Wang, J.: A precise and efficient evaluation of the proximity between web clients and their local DNS servers. In: Proceedings of the USENIX Annual Technical Conference (2002)
28. Frank, B., Poese, I., Smaragdakis, G., Uhlig, S., Feldmann, A.: Content-aware traffic engineering. In: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems, pp. 413–414 (2012). DOI 10.1145/2254756.2254819
29. Liu, X., Dobrian, F., Milner, H., Jiang, J., Sekar, V., Stoica, I., Zhang, H.: A case for a coordinated internet video control plane. In: Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 359–370 (2012). DOI 10.1145/2342356.2342431
30. Niven-Jenkins, B., Le Faucheur, F., Bitar, N.: Content distribution network interconnection (CDNI) problem statement. RFC 6707 (Informational) (2012). URL `http://www.ietf.org/rfc/rfc6707.txt`

31. ETSI: CDN interconnection architecture. ETSI TS 182 032 (2013). URL http://www.etsi.org/deliver/etsi_ts/182000_182099/182032/01.01.01_60/ts_182032v010101p.pdf
32. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. SIGCOMM Computer Communications Review **38**(2), 69–74 (2008). DOI 10.1145/1355734.1355746
33. Chowdhury, N.M.K., Boutaba, R.: Network virtualization: State of the art and research challenges. IEEE Communications Magazine **47**(7), 20–26 (2009). DOI 10.1109/MCOM.2009.5183468
34. Farrel, A., Ayyangar, A., Vasseur, J.: Inter-domain MPLS and GMPLS traffic engineering – resource reservation protocol-traffic engineering (RSVP-TE) extensions. RFC 5151 (Proposed Standard) (2008). URL http://www.ietf.org/rfc/rfc5151.txt
35. Kuipers, F., Van Mieghem, P., Korkmaz, T., Krunz, M.: An overview of constraint-based path selection algorithms for QoS routing. IEEE Communications Magazine **40**(12), 50–55 (2002). DOI 10.1109/MCOM.2002.1106159
36. Xue, G., Zhang, W., Tang, J., Thulasiraman, K.: Polynomial time approximation algorithms for multi-constrained QoS routing. IEEE/ACM Transactions on Networking **16**(3), 656–669 (2008). DOI 10.1109/TNET.2007.900712
37. Todd, M.J.: The many facets of linear programming. Mathematical Programming **91**(3), 417–436 (2002). DOI 10.1007/s101070100261
38. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.Y., Moon, S.: I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pp. 1–14 (2007). DOI 10.1145/1298306.1298309
39. Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D., Mahanti, A.: Characterizing web-based video sharing workloads. ACM Transactions on the Web **5**(2), 1–27 (2011). DOI 10.1145/1961659.1961662
40. Avramova, Z., Wittevrongel, S., Bruneel, H., De Vleeschauwer, D.: Analysis and modeling of video popularity evolution in various online video content systems: Power-law versus exponential decay. In: Proceedings of the First International Conference on Evolving Internet (INTERNET), pp. 95–100 (2009)
41. Wu, T., Timmers, M., De Vleeschauwer, D., Van Leekwijck, W.: On the use of reservoir computing in popularity prediction. In: Proceedings of the Second International Conference on Evolving Internet (INTERNET), pp. 19–24 (2010). DOI 10.1109/INTERNET.2010.13
42. Garroppo, R.G., Giordano, S., Tavanti, L.: A survey on multi-constrained optimal path computation: Exact and approximate algorithms. Computer Networks **54**(17), 3081–3107 (2010). DOI 10.1016/j.comnet.2010.05.017
43. Karmarkar, N.: A new polynomial time algorithm for linear programming. Combinatorica **4**(4), 373–395 (1984)
44. Dash, S.: An exponential lower bound on the length of some classes of branch-and-cut proofs. In: Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization, pp. 145–160 (2002)
45. Achterberg, T., Berthold, T.: Improving the feasibility pump. Discrete Optimization **4**, 77–86 (2007). DOI 10.1016/j.disopt.2006.10.004
46. Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS Journal on Computing **19**(4), 534–541 (2007). DOI 10.1287/ijoc.1060.0189
47. Gamer, T., Scharf, M.: Realistic simulation environments for IP-based networks. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (Simutools) (2008)
48. Tang, W., Fu, Y., Cherkasova, L., Vahdat, A.: Modeling and generating realistic streaming media server workloads. Computer Networks **51**(1), 336–356 (2007). DOI 10.1016/j.comnet.2006.05.003
49. Monath, T., Kind, M., Heger, T., Schlesinger, M., Aznar, J.: Economical analysis of experience-optimized service delivery. In: Proceedings of the 9th Conference on Telecommunications, Internet and Media Techno Economics (CTTE) (2010). DOI 10.1109/CTTE.2010.5557711
50. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable video coding extension of the H.264/AVC standard. IEEE Transactions on Circuits and Systems for Video Technology **17**(9), 1103–1120 (2007). DOI 10.1109/TCSVT.2007.905532
51. Yu, J., Chou, T., Yang, Z., Du, X., Wang, T.: A dynamic caching algorithm based on internal popularity distribution of streaming media. Multimedia Systems **12**(2), 135–149 (2006). DOI 10.1007/s00530-006-0045-x
52. Chen, S., Shen, B., Wee, S., Zhang, X.: Segment-based streaming media proxy: Modeling and optimization. IEEE Transactions on Multimedia **8**(2), 243–256 (2006). DOI 10.1109/TMM.2005.864281

Jeroen Famaey obtained a masters degree in computer science from Ghent University, Belgium, in June 2007. He received his Ph.D. degree, on federated and autonomic management

of multimedia services, from the same university in June 2012. Since then he has been working as a post-doctoral researcher in the area of network and service management, affiliated with Ghent University and iMinds. Within this area, he has (co-)published more than 30 articles in peer-reviewed international journals and conference proceedings.

Steven Latré is an assistant professor at the University of Antwerp, Belgium and the Future Internet Department at iMinds. He received a Master of Science degree in computer science from Ghent University, Belgium and a Ph.D. in Computer Science Engineering from the same university. His research activity focuses on autonomous management and control of both networking and computing applications.

Tim Wauters received his M.Sc. and Ph.D degrees in electro-technical engineering in June 2001 and January 2007 from Ghent University, Belgium, where he is active as a post-doctoral fellow of the F.W.O.-V at the Department of Information Technology (INTEC). His research focuses on network and service management solutions for scalable multimedia delivery.

Filip De Turck is a professor and leads the network and service management research group at the Department of Information Technology of the Ghent University, Belgium and the Future Internet Department of the iMinds research center, Flanders. He is a senior IEEE member and author or co-author of more than 350 refereed papers published in international journals or in the proceedings of international conferences in the area of network and service management and design of new communication services.