Volume 29, issue 16, 12 October 2006
ISSN 0140-3664
ELSEVIER
computer
communications
Available online at
ScienceDirect
www.sciencedirect.com
Special Issue:
More than a Protocol for Next Generation Internet
Guest Editors: Jiann-Liang Chen
Han-Chieh Chao
Sy-Yen Kuo

Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 29 (2006) 3313–3326
computer communications
www.elsevier.com/locate/comcom

Replica placement in ring based content delivery networks

Tim Wauters *, Jan Coppens, Filip De Turck, Bart Dhoedt, Piet Demeester

Department of Information Technology (INTEC), Ghent University – IMEC – IBBT, Gaston Crommenlaan 8, bus 201, B-9050 Ghent, Belgium

Received 22 February 2006; received in revised form 12 May 2006; accepted 15 May 2006
Available online 12 June 2006

Abstract

The recent introduction of Content Distribution Networks (CDNs) enhances the delivery of high quality multimedia content to end users. In a CDN architecture, the content is replicated to so-called surrogate servers, generally at the edge of the transport network, to improve the quality of service (QoS) of streaming multimedia delivery services. By using peer-to-peer (P2P) technologies, these edge servers can co-operate and provide a more scalable and robust service in a self-organizing CDN.

In this paper, we propose a set of distributed replica placement algorithms (RPAs), based on an Integer Linear Programming (ILP) formulation of the centralized content placement problem. These algorithms further enhance the CDN performance by optimizing the network and server load, reducing network delays and avoiding congestion. Although the proposed algorithms are designed for and tested on different network topologies, we focus on robust ring based CDNs in this study. Content placement on such a network topology can be calculated analytically and can be used for comparison.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Content distribution network; Replica placement algorithm; Peer-to-peer

1. Introduction

The quality of service offered by single server (or single site cluster) architectures is often insufficient for popular multimedia websites or streaming services. Overloaded servers and congested transport networks degrade the user experience noticeably. A growing number of content providers therefore benefits from content distribution services offered by companies like Akamai [1]. They use high capacity overlay networks in combination with surrogate servers at the edge of the Internet to deliver their bandwidth-intensive content. Consequently, the central server is offloaded, and the latency and network traffic reduced.

The development of static (*offline*) placement strategies for server replicas further enhances CDN performance [4–7,13]. These algorithms decide where to replicate specific content, in order to reduce bandwidth consumption and latency at low infrastructure usage costs. Content Server Selection algorithms then direct users to the appropriate surrogate server, offering the best achievable quality of service.

Inspired by the extremely popular peer-to-peer file sharing applications, P2P technologies at the level of the surrogate servers have been introduced. This makes direct communication between them possible, so that information on local traffic patterns can be exchanged. By using these P2P architectures in CDNs [8], a more robust, scalable and efficient service can be provided. The distributed replica placement and retrieval algorithms, to be executed by all cache nodes independently, can now dynamically (*online*) adapt to new content supplied to the CDN, to a changing user request pattern or to varying network conditions.

This paper studies offline as well as online replica placement strategies, applicable on general network topologies. In order to be able to compare both approaches to an analytical model, the presented experiments are performed on ring topologies, which are widely used in recent CDN deployments. The efficiency of the proposed RPAs has also

* Corresponding author. Tel.: +32 9 33 14977; fax: 32 9 33 14899.
E-mail addresses: tim.wauters@intec.ugent.be (T. Wauters), jan.coppens@intec.ugent.be (J. Coppens), filip.deturck@intec.ugent.be (F. De Turck), bart.dhoedt@intec.ugent.be (B. Dhoedt), piet.demeester@intec.ugent.be (P. Demeester).

been validated on more general topologies, which has been presented in other work [9,17]. The simulations for the centralized ILP solution have been performed using Cplex [19], the distributed algorithms have been evaluated using a discrete event simulator.

The remainder of this paper is divided into two main parts: Sections 3–5 present a centralized and static (*offline*) approach for the replica placement problem, while Sections 6 and 7 propose a distributed and dynamic (*online*) replica placement strategy.

Into more detail, this paper is structured as follows. First an overview of related work on replica placement in CDNs is presented in Section 2. Section 3 introduces an analytical solution technique and an ILP formulation for the centralized approach on networks without topology constraints. The main costs to be minimized are the overall network bandwidth consumption and storage costs. Both models are compared for a ring based CDN topology in Section 4. An approximation of the analytical solution will lead to basic design rules for ring based CDNs. The analytical solution can also be used for a larger network topology, including a tree-based access network, as studied in Section 5.

Two distributed heuristics are proposed in Section 6. Contrary to the computationally heavy centralized solutions (this problem is NP-complete, as proven in [7]), these algorithms can be executed on networks with more complex topologies. Furthermore, they are able to dynamically adapt the replica placement to changing user demands, varying network occupations or new content added to the network. Simulation results for these heuristics on a network with a ring topology are compared to results from the static ILP formulation. In Section 7, these distributed algorithms are extended to dynamically support load balancing and avoid congestion in the network.

The last section concludes this paper and presents some ideas for our future work, as this study is part of a more general content distribution architecture [15,16].

2. Related work

The advantages of replica placement algorithms over typical caching strategies have been studied in [5] and various RPAs have been proposed in recent studies [4,13]. A detailed overview of the available models and algorithms as well as a framework for evaluating them is given in [6]. While most models have a similar cost function (optimizing bandwidth and/or storage usage costs for a given request pattern), less attention has been given to network constraints (limited link or storage capacities). Furthermore, a large part of these algorithms are designed for specific network topologies only (e.g. tree topology [14]). The possible use of these algorithms to reduce the server load or to avoid network congestion has not been given much consideration either. The benefits of adding workload information to placement algorithms are studied in [4]. Although [7] and [8] show that the introduction of peer-to-peer systems in content delivery networks has a potential to further improve the network performance, few developments have been made on distributed replica placement algorithms. These studies on RPAs in CDNs [4,11,12] and unicast streaming CDNs [10], as well as similar work on proxy caching techniques [18] also show that greedy algorithms that take distance metrics and content popularity into account perform better than more straightforward heuristics such as *LRU* (Least Recently Used) or *LFU* (Least Frequently Used).

This work deals with the aforementioned shortcomings, in order to offer a replica placement solution that enhances the service quality at low network costs, for different topologies.

3. General static problem formulation

As stated in the introduction of this paper, we start with handling the static problem, i.e. the content delivery system is in steady state, from a centralized point of view. This means that the set of requested objects (streams) does not change, and the request rates of these objects also remain constant. We start with the analytical formulation for general network topologies and present an ILP formulation that takes network constraints into account afterwards.

3.1. Analytical formulation

Let $\mathbf{O} = \{o_1,\ldots,o_F\}$ denote the object set which is offered for download to the users, and let r_i denote the total number of requests for object o_i during the period $[0, T]$. The size (measured in bytes) of object o_i equals s_i, the streaming bitrate is b_i. The infrastructure to host the content set $\mathbf{O}$ is characterized by a graph $\mathbf{G}$, consisting of a set of vertices $\mathbf{V}$ (of size N), which are interconnected by a set of edges $\mathbf{E}$. For the time being, we assume that the edges are not congested and able to carry the traffic generated in the content delivery network.

Given $\mathbf{G}$, $\mathbf{O}$, r_i, s_i and b_i, the problem now is to find the optimal set of surrogate servers $\mathbf{S}_i$ (with cardinality $|\mathbf{S}_i| = n_i$) for each object o_i by optimizing the cost associated with both transmitting and storing this particular object at the surrogate servers. When we define C_T as the cost to transmit one unit (e.g. one object) over one link and C_S as the cost to store one unit of content (e.g. one object), the input parameter $\alpha = C_S/C_T$ indicates the relative trade-off between storage and transport costs. Without loss of generality, C_T can be set to one, so that the cost C_i, incurred by storing and streaming the object o_i in the CDN, is then given by

$$C_i = b_i r_i \overline{d_i}(\mathbf{S}_i) + \alpha s_i n_i, \quad (1)$$

where $\overline{d_i}$ is the average distance between requestor and node serving the request. This average distance can either be a simple hop count or a more sophisticated sum of individual link costs. Obviously, as suggested in Eq. (1), the quantity $\overline{d_i}$ is a function of the set $\mathbf{S_i}$. The contribution

$b_i r_i \overline{d_i(\mathbf{S}_i)}$ to (1) will be referred to as *transport cost*, while $\alpha s_i n_i$ will be called *storage cost*.

In principle, since the server disk capacity is assumed unlimited, as well as link bandwidth, the total cost

$$C = \sum_{i=1}^{F} C_i \tag{2}$$

can be optimized by minimizing each of the C_i independently. This minimization can be done straightforwardly by an exhaustive strategy. However, it is clear that solving the problem this way becomes computationally unfeasible for large values of N, since C_i should be calculated for each subset of $\mathbf{V}$ (excluding of course the empty set, since each object should be available at at least one location, so $n_i > 0$). However, one can easily show $C_i(n_i)$ to have a single minimum (because the transmission part monotonically decreases as a function of n_i, while the storage contribution obviously increases), and therefore, when considering increasing values of n_i (starting with $n_i = 1$), the search comes to an end as soon as C_i increases. This all leads to the rather simple algorithm to calculate the optimal surrogate server location sets $\mathbf{S_i}$ and associated minimal costs presented in Fig. 1. Based on the results found using this procedure, the CDN can be dimensioned (surrogate server sizes and link bandwidth). Of course, if not all nodes of the physical network are eligible for storing content, one removes these nodes from the set $\mathbf{V}$, ensuring that the value $\overline{d_i}$ is calculated correctly.

In general, solving the CDN optimization problem using this procedure is complex, mainly due to the complicated structure of the function $\overline{d_i(\mathbf{S_i})}$, which both depends on network topology ($\mathbf{G}$) and request patterns (request rates from each end user location for each file). For regular topologies and request patterns, and more specifically ring networks with uniform user behavior, it will be shown in Section 4 that the optimization problem can be solved analytically.

```
1.    For each $i = 1 .. F$
1.1       $C_i(0) =$ , $n_i=1$
1.2       Find $\mathbf{S_i}(n_i)$ yielding minimal cost, i.e.
              $C_i(n_i) = \min_{\substack{\#S_i(n_i)=n_i \\ S_i(n_i)\subset V}} b_i r_i \overline{d_i(\mathbf{S_i}(n_i))} + \alpha s_i n_i$
1.3       If $n_i=N$ or $\Delta^- C_i(n_i) = C_i(n_i) - C_i(n_i - 1) > 0$
              Let $\mathbf{S_i} = \mathbf{S_i}(n_i)$
          Else, increase $n_i$ by 1 and repeat step 1.2
2.    Optimal surrogate server locations $\mathbf{S_i}$ found for all i=1 .. $F$, calculate
      minimal cost from
          $C = \sum_{i=1}^{F} \left(b_i r_i \overline{d_i(\mathbf{S_i})} + \alpha s_i n_i\right)$
```

Fig. 1. Exhaustive strategy to calculate optimal surrogate server location sets for replica placement for a set of objects O=$\{o_1,\ldots,o_F\}$, characterized by request rates r_i.

3.2. ILP-problem formulation

The algorithm shown in Fig. 1 has several drawbacks, besides of being computationally intensive for large networks. No limitations on surrogate server sizes, nor on bandwidth usage are taken into account, and hence the procedure is not suited for optimizing resource usage on an already installed infrastructure. To overcome these shortcomings, an Integer Linear Programming (ILP) formulation is presented in this section.

3.2.1. Network parameters

Every edge e from $\mathbf{E}$ has a cost parameter c_e (e.g. a delay penalty to model congestion on the link) and a maximum bandwidth capacity u_e. All nodes n from $\mathbf{V}$ have a storage capacity m_n (higher than 0 for the cache nodes $\mathbf{A} \subset \mathbf{V}$). The cost to store an object o from $\mathbf{O}$ is equal to the size s_o of the object. Apart from a size s_o, every object also has a fixed bitrate b_o. This may correspond to the constant bit rate of a streaming file for a Video on Demand service. The requests rates $r_{n,o}$ from the user nodes n from $\mathbf{D} \subset \mathbf{V}$ for each of the files are given. These rates also reflect the popularity of the files to the users. We also define $\mathbf{D}_o$ as all the users requesting object o.

The main variables in the objective function are the transport variables $h_{e,d,o}$ and the storage variables $z_{n,o}$:

- $h_{e,d,o}$ is 1 if edge e is used to deliver object o to destination d, 0 otherwise
- $z_{n,o}$ is 1 if node n is used to cache object o, 0 otherwise

There is one auxiliary variable $x_{n,d,o}$:

- $x_{n,d,o}$ is 1 if node n is used to cache object o for destination d, 0 otherwise

We define I_n as the set of incoming edges of node n, O_n as the set of outgoing edges.

3.2.2. Objective function

Now that all symbols and variables are explained, the objective function F can be expressed as follows:

$$F = \sum_{d\in D_o} \sum_{o\in O} \sum_{e\in E} c_e b_o r_{d,o} h_{e,d,o} + \alpha \sum_{o\in O} \sum_{n\in V} s_o z_{n,o}. \tag{3}$$

The objective function has to be minimized and consists of two parts, the *transport cost* and the *storage cost*:

- The first part of formula (3) is the *transport cost*. It is the cost related to the use of bandwidth. If edge e is used to transport object o to destination node d (or in other words if $h_{e,d,o}$ is 1) then there is a cost of $c_e b_o r_{d,o}$ associated with that use.
- The second part is the *storage cost*. It defines the cost for caching object o in node n as s_o (the size of object o), if $z_{n,o}$ is 1.

In order to be able to emphasize on the importance of one of these costs, a parameter α is introduced to linearly combine both costs. If α is zero, only the transport cost is considered to be important. If α is high the storage cost is more important and the solution found will have only few cached files.

3.2.3. Constraints

3.2.3.1. Capacity constraints.

$$\sum_{d\in D_o}\sum_{o\in O} b_o h_{e,d,o} \leqslant u_e \quad \forall e \in E, \tag{4}$$

$$\sum_{o\in O} s_o z_{n,o} \leqslant m_n \quad \forall n \in V. \tag{5}$$

Constraint (4) imposes a restriction on the total flow through the edges. This flow cannot exceed the capacity of the edge. Constraint (5) imposes a restriction on the amount of cached content in a certain node. This cost must not exceed the capacity of that node. There are $|\mathbf{E}| + |\mathbf{V}|$ capacity constraints.

3.2.3.2. Auxiliary constraints.

$$x_{n,d,o} \leqslant z_{n,o} \quad \forall d \in D_o, \quad \forall o \in O, \quad \forall n \in V. \tag{6}$$

This auxiliary constraint takes care of the relationship between $x_{n,d,o}$ and $z_{n,o}$. Constraint (6) indicates that if node n stores object o (i.e. $z_{n,o}$ is one), this can be done for multiple destinations (i.e. $x_{n,d,o}$ can be one for several destination nodes $\mathrm{o} \in \mathrm{D}_o$).

3.2.3.3. Flow conservation constraints.

$$\sum_{e\in I_n} h_{e,d,o} = \sum_{e\in O_n} h_{e,d,o} \quad \forall n \in V \setminus d \setminus A, \quad \forall d \in D_o, \quad \forall o \in O, \tag{7}$$

$$x_{n,d,o} + \sum_{e\in I_n} h_{e,d,o} = \sum_{e\in O_n} h_{e,d,o} \quad \forall n \in A, \quad \forall d \in D_o, \quad \forall o \in O, \tag{8}$$

$$\sum_{e\in I_n} h_{e,d,o} = 1 \quad \forall n \in D_o, d = n, \forall o \in O, \tag{9}$$

$$\sum_{e\in I_n} h_{e,d,o} = \sum_{e\in O_n} h_{e,d,o} \quad \forall n \in D, \quad \forall d \in D_o,\ d \neq n, \quad \forall o \in O. \tag{7$'$}$$

These constraints regulate the flows between source and destination nodes. Constraint (7) ensures the traffic through "normal" nodes (nodes that are not cache or destination nodes), constraint (8) takes care of cache nodes and constraint (9) is for destination nodes.

Constraint (7) indicates that node n should let incoming data from object o for destination d pass through to the next node on the path. Constraint (7′) does the same for destination nodes acting as normal nodes (e.g. destination nodes laying on the path towards other destination nodes). Constraint (8) indicates that cache node n should let incoming data from object o for destination d pass through to the next node on the path, except when it is the source node for that download (then $x_{n,d,o} = 1$ and $h_{e,d,o} = 0$ on all incoming edges). Constraint (9) indicates that destination node n should receive the object he requested on one of his incoming edges.

3.2.3.4. Binary constraints.

$$h_{e,d,o}\text{binary} \quad \forall e \in E, \ \forall d \in D_o, \quad \forall o \in O, \tag{10}$$

$$z_{n,o}\text{binary} \quad \forall n \in V, \quad \forall o \in O, \tag{10$'$}$$

$$x_{n,d,o}\text{binary} \quad \forall n \in V, \quad \forall d \in D_o, \quad \forall o \in O. \tag{10$''$}$$

Constraint (10) imposes that all variables are binary.

3.2.3.5. Additional constraint.

$$\sum_{e\in E} h_{e,d,o} p_e \leqslant p_{\max} \quad \forall d \in D_o, \quad \forall o \in O. \tag{11}$$

Constraint (11) can be used as an additional constraint to set a maximum penalty on a parameter for each object stream. Examples are restrictions on the total delay (p_e represents the delay on link e) or the hopcount ($p_e = 1$), for the total path of a stream.

3.2.3.6. Robustness constraint.

$$\sum_{n\in A} z_{n,o} \geqslant f + 1 \quad \forall o \in O. \tag{12}$$

Restriction (12) adds robustness to the content delivery service. Every file should at least have $f + 1$ different locations in the network, with f the maximum number of simultaneously failing caches.

4. Network design for ring based CDNs

In this section, we make a comparison between the analytical and the ILP model for a CDN with a ring topology and determine a set of network design rules. The ring network consists of N nodes that are all candidate surrogate servers (Fig. 2). A total of F objects $\{o_1, \ldots, o_F\}$ are available for a certain period $[0, T]$. During that period, a total of R requests are made, with r_i requests for object o_i. We make the additional assumption that all users are connected through an access network link to one of the N ring nodes. Since this access network is assumed given, the transport cost on these access links cannot be optimized. Furthermore, we assume that all requests are equally spread over the N nodes during the given time interval.

Since the number of possible replica placements or routes is limited on a ring network, the centralized approach is still scalable for larger values of N. On more complex topologies, the scalability of the ILP solution is very limited (outside of the scope of this paper, studied in [9,17]). The distributed solution presented later on in this paper however is very scalable, since only local traffic patterns are taken into account.

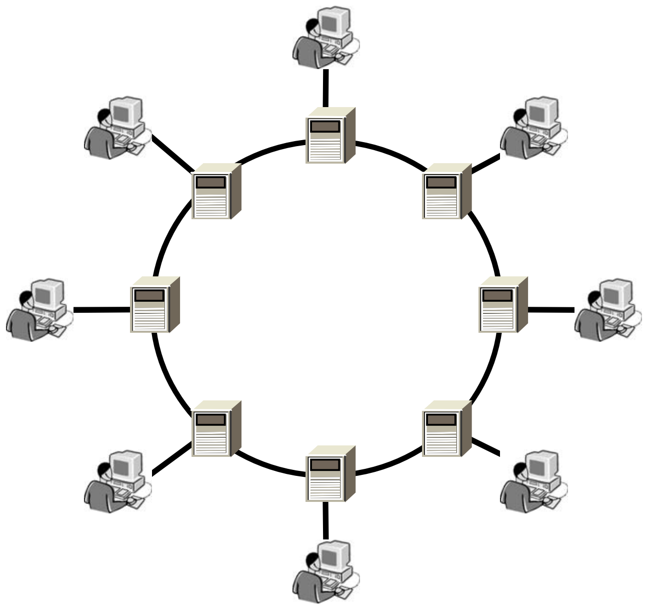


Fig. 2. Ring network with access network links.

4.1. Analytical solution

Due to the symmetry of the problem, it is obvious that for a given number of surrogate servers, an optimal location is achieved by maximizing the distance between individual surrogate servers. For N nodes and n surrogate servers, and q the remainder of dividing N by n, such that

$$N = \left\lfloor \frac{N}{n} \right\rfloor n + q = d_0 n + q, \tag{13}$$

we have $n - q$ nodes serving requests aggregated by d_0 ring nodes, and q nodes serving $d_0 + 1$ nodes (of course assuming shortest path routing). Note that this observation yields an optimal set $\mathbf{S_i}$ for a given n_i, thereby avoiding the optimization step of the algorithm presented in Fig. 1 (more specifically step 1.2). The following analytical expression for $\overline{d_i(n)}$ can easily be derived:

$$\overline{d_i(n)} = \left\lfloor \frac{d_0 + 1}{2} \right\rfloor \left(1 - \frac{n}{N} \left\lfloor \frac{d_0 + 1}{2} \right\rfloor \right). \tag{14}$$

Given this expression, the procedure given in Fig. 1 can be considerably simplified as follows (see Fig. 3), observing that

$$\begin{aligned} \Delta^- C_i(n_i) &= C_i(n_i) - C_i(n_i - 1) > 0 \\ &= b_i r_i \left[\overline{d_i(n_i)} - \overline{d_i(n_i - 1)} \right] + \alpha s_i \\ &= b_i r_i \Delta^- \overline{d_i(n_i)} + \alpha s_i \end{aligned} \tag{15}$$

Note that this procedure is valid for *any* problem (topology and request pattern) where $\overline{d_i(n)}$ is only a function of the size of $\mathbf{S_i}$, and where the optimal surrogate server location $\mathbf{S_i}$ can be found directly from n_i. This is for example not the case when the user demand is asymmetrical. In that case, this analytical solution offers an upper limit for the total cost, since the replicas of an object can be placed closer

```
1   For each $i = 1 .. F$
        Calculate $n_i = \max_{\substack{\Delta^- C_i(n) > 0 \\ 0 < n \le N}} n$
2   Optimal server locations found for all i=1 .. $F$, calculate minimal
    cost from
        $C = \sum_{i=1}^{F} \left( b_i r_i \overline{d_i(n_i)} + \alpha s_i n_i \right)$
```

Fig. 3. Exhaustive strategy to calculate optimal surrogate server location sets for replica placement for a set of objects $O = \{o_1, \ldots, o_F\}$, characterized by request rates r_i, on a ring based CDN.

to the users requesting that object than for a symmetrical user demand.

In Fig. 4 this analytical solution is compared to the ILP solution for a ring network with $N = 8$ surrogate servers, serving a total of $R = 10{,}000$ requests for $F = 20$ objects. Small differences in transport and storage cost are visible, but the total cost (transport cost + $\alpha \cdot$ storage cost) is identical in both cases. When α is sufficiently low (low storage cost), all 20 objects are replicated on all surrogate servers (storage cost = $F \cdot N = 160$ units) and the transport cost is limited to the fixed streaming cost on the access links (transport cost = $R \cdot \mathrm{d}(n) = 10{,}000$ units, since the average distance $d(n)$ on the access network is 1). When storage is expensive (α is high), every object is found on only one surrogate server (storage cost = $F \cdot 1 = 20$ units) and the transport cost reaches its maximum value (transport cost = $R \cdot d(n) = 30{,}000$ units, since the average distance $d(n)$ is 3:1 for the access network link plus 2 for the average distance on the ring network with 8 surrogate servers). Note that the transport and storage costs are much more sensitive to small changes to the input parameter α for lower values ($\alpha < 100$).

An approximation n_i' to the optimal value n_i can be found by solving

$$\Delta^- C_i(n_i') = 0 \tag{16}$$

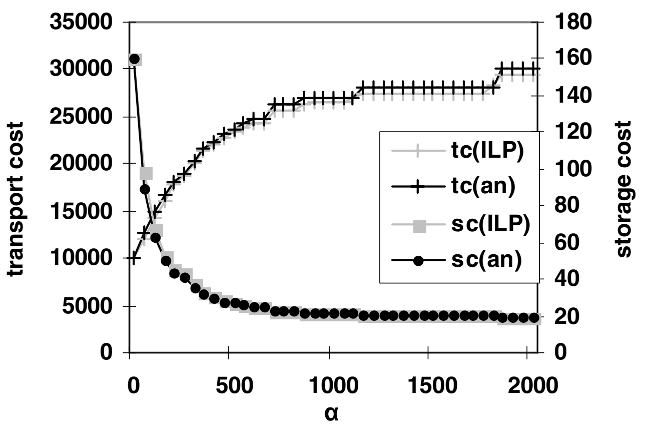


Fig. 4. Transport (tc) and storage (sc) cost in a ring network with 8 surrogate servers (20 files available), for both the analytical and the ILP solution.

in the interval $[1, N]$. If (16) has no solution in this interval, either $n_i' = 1$ or $n_i' = N$, depending on the sign of $\Delta^- C_i(1)$. When we assume that each object has the same size s_i and streaming bandwidth b_i, we can set s_i and b_i to one without loss of generality. Solving (16) is then clearly equivalent to solving

$$\Delta^- \overline{d_i(n_i')} = -\frac{\alpha}{r_i}. \tag{17}$$

In the specific case of a ring based CDN, we can find an additional estimate n_i'' for n_i, by approximating the function $\overline{d_i(n)}$ Eq. (14) by

$$\overline{d_i'(n)} = \frac{N^2 - n^2}{4Nn}, \tag{18}$$

which is obtained by replacing all integer divisions by their real valued counterparts. This allows to find

$$\Delta^- \overline{d_i'(n)} = -\frac{1}{4N} - \frac{N}{4n(n-1)} \tag{19}$$

and solving now

$$\Delta^- \overline{d_i'}(n_i'') = -\frac{\alpha}{r_i} \tag{20}$$

(which is a quadratic equation in n_i'') gives the following approximation for n_i:

$$\frac{1}{2}\left[1 + \sqrt{1 + \left[\frac{\alpha}{Nr_i} - \frac{1}{4N^2}\right]^{-1}}\right]. \tag{21}$$

Since however n_i should be an integer value, satisfying $\Delta^- C_i(n_i) < 0$, and because approximation (21) implies $\Delta^- C_i(n_i) = 0$, we expect the approximation (21) to be systematically too large. More specifically, since we should round (21) down to the smaller integer value, this estimate is on average too large by 0.5, giving the following improved estimate for n_i

$$n_i'' = \frac{1}{2}\sqrt{1 + \left[\frac{\alpha}{Nr_i} - \frac{1}{4N^2}\right]^{-1}}. \tag{22}$$

One further notices that apparently the optimal number of surrogate servers for each object is strongly dependent on the parameter $\frac{\alpha}{Nr_i}$, and to a lesser degree the network size N itself. If the expression in the right hand side of Eq. (21) gives results exceeding N, of course the limit value N is taken as estimate for n_i. Similarly, since at least one copy of each object should be stored in the network, the value one is taken as approximation for n_i in case (22) yields values smaller than one. More explicitly

$$n_i'' = \begin{cases} N & \frac{\alpha}{Nr_i} < \frac{1}{4N^2} + \frac{1}{4N^2-1} \\ 1 & \frac{\alpha}{Nr_i} > \frac{1}{4N^2} + \frac{1}{3} \\ \frac{1}{2}\sqrt{1 + \left[\frac{\alpha}{Nr_i} - \frac{1}{4N^2}\right]^{-1}} & \text{otherwise} \end{cases}. \tag{23}$$

4.2. Design rules for ring based CDNs

In this section, the results obtained above for pure ring based CDNs are used to dimension both storage space and network capacity. To arrive at numerical results for these values, assumptions must be made concerning the relative request rates of the objects in the set **O**. The well-accepted Zipf-like distribution [2,3] is here used to describe this relative object popularity, i.e.

$$r_i \propto \frac{1}{i^\beta}, \quad 1 \leqslant i \leqslant F \tag{24}$$

with typical values for β between 0.5 and 1.0 [2,3]. Let the total amount of requests (i.e. requests from all users during the interval $[0, T]$) be R, giving

$$r_i = R\frac{i^{-\beta}}{\sum_{i=1}^{F} i^{-\beta}} = \frac{R}{A} i^{-\beta}, \quad 1 \leqslant i \leqslant F. \tag{25}$$

Large β values indicate a relatively small set of extremely popular objects, leading to less storage space requirements at the surrogate servers.

4.2.1. Storage capacity

The total storage capacity needed in the ring network (s), in case the total cost is optimized, can be calculated from

$$s = \sum_{i=1}^{F} s_i n_i. \tag{26}$$

Assuming no correlation between object size and popularity, and denoting the average object size as $\bar{s}$, this becomes

$$s = \bar{s}\sum_{i=1}^{F} n_i \approx \bar{s}\sum_{i=1}^{F} n_i''. \tag{27}$$

In order to calculate the latter value (where the approximation (23) is used for n_i), the object indices i_1 and i_N are derived from (23). The index i_N is the largest value for which n_i'' yields the value N, while i_1 is the smallest value for which only one object copy is stored in the ring. From (23) it follows that

$$\begin{cases} \frac{\alpha}{Nr_{i_1}} = \frac{1}{4N^2} + \frac{1}{3} \\ \frac{\alpha}{Nr_{i_N}} = \frac{1}{4N^2} + \frac{1}{4N^2-1}, \end{cases} \tag{28}$$

which, using the Zipf-like popularity distribution (25) yields immediately

$$\begin{cases} i_1 = \left[\frac{NR}{\alpha A}\left(\frac{1}{4N^2} + \frac{1}{3}\right)\right]^{1/\beta} \\ i_N = \left[\frac{NR}{\alpha A}\left(\frac{1}{4N^2} + \frac{1}{4N^2-1}\right)\right]^{1/\beta}. \end{cases} \tag{29}$$

Using these values, s can now be calculated as

$$s \approx \bar{s}\sum_{i=1}^{F} n''_i = \bar{s}\left(N\min(i_N,F) + \sum_{\max[\min(i_N F),1]}^{\min(i_1,F)} \tfrac{1}{2}\sqrt{1+\left[\tfrac{\alpha A}{NR}i^{\beta}-\tfrac{1}{4N^2}\right]^{-1}} + max(F-i_1,0)\right). \quad (30)$$

To simplify this expression, the middle term (30) is replaced by

$$\sum_{\max[\min(i_N F),1]}^{\min(i_1,F)} \frac{1}{2}\sqrt{\frac{NR}{\alpha A}i^{-\beta}}$$

which is justified in view of the *i*-range values (between i_1 and i_N) of interest for this expression. If now the summations are approximated by integrals, we find the following expression for *s*:

$$\frac{s}{\bar{s}} \approx N\min(i_N,F) + \frac{1}{2}\sqrt{\frac{NR}{\alpha}}\sqrt{\frac{1-\beta}{F^{1-\beta}-1}} \times \frac{\min(i_1,F)^{1-\beta/2} - \max[\min(i_N,F),1]^{1-\beta/2}}{1-\beta/2} + \max(F-i_1,0). \quad (31)$$

Of course, the case $i_N > F$ is of no practical use, since this would imply that all objects are stored on all locations, and that the ring network is actually not used. Therefore, for all practical situations, (31) becomes

$$\frac{s}{\bar{s}} \approx Ni_N + \frac{1}{2}\sqrt{\frac{NR}{\alpha}}\sqrt{\frac{1-\beta}{F^{1-\beta}-1}} \times \frac{\min(i_1,F)^{1-\beta/2} - \max(i_N,1)^{1-\beta/2}}{1-\beta/2} + \max(F-i_1,0). \quad (32)$$

4.2.2. Link capacity

The total link capacity needed in the ring network (*l*), in case the total cost is optimized, can be calculated from

$$l = \sum_{i=1}^{F} b_i r_i \overline{d_i(n_i)}. \quad (33)$$

Assuming no correlation between object bitrate and popularity, and denoting the average object bitrate as $\bar{b}$, this becomes

$$l = \bar{b}\sum_{i=1}^{F} r_i\overline{d_i(n_i)} \approx \bar{b}\sum_{i=1}^{F} r_i\overline{d'_i(n_i)}. \quad (34)$$

Taking into account the approximation (18) and the Zipf-like distribution (25), the total link capacity is given by

$$\frac{l}{\bar{b}} = \sum_{i=1}^{F} \frac{R}{A} i^{-\beta} \frac{N^2 - n_i^2}{4Nn_i}. \quad (35)$$

Fig. 5 compares the transport and storage cost for the exact and approximated analytical solution. The curves for the storage cost *s* and the transport cost *l* are given by the Eqs. (31) and (35) respectively.

4.2.3. Influence of traffic parameters

We used the analytical solution to study the influence of different traffic parameters on the transport cost for a given network design. We assume that a ring network with 8 surrogate servers is optimally designed for distributing 20 files with a Zipf-like content popularity [2] (Fig. 6) with parameter $\beta = 0.7$ (according to [2] and our own measurements on peer-to-peer file sharing applications [3]). In total 10,000 requests are made for these files, evenly distributed over the 8 surrogate servers. The transport cost for the scenario with symmetrical user demand is given in Fig. 7 (*tc (1:1)*,(*x*:*y*) meaning that for every *x* requests at the first surrogate server, *y* requests are made at each other surrogate server).

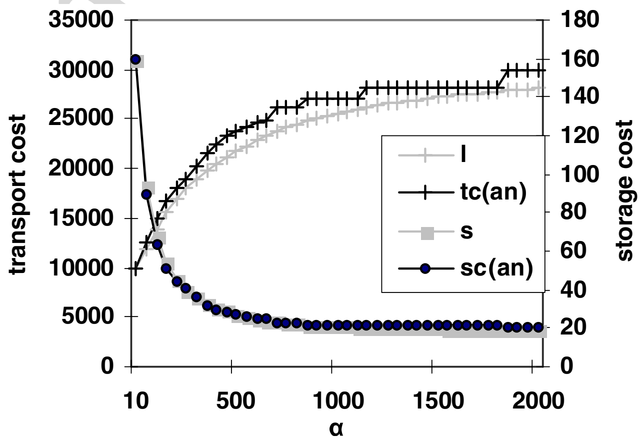


Fig. 5. Transport (tc) and storage (sc) cost in a ring network with 8 surrogate servers, for both the exact and the approximated solution.

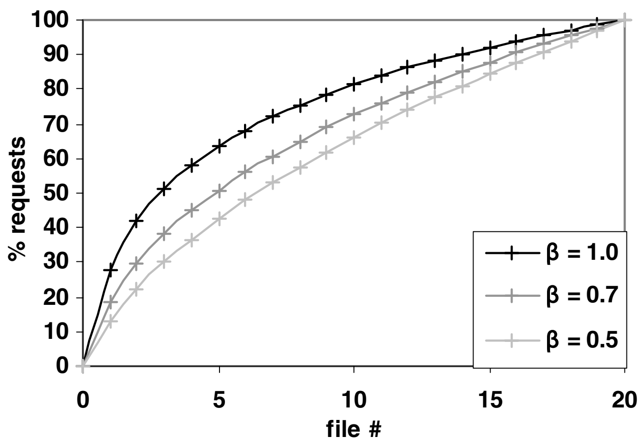


Fig. 6. Cumulative Zipf-like distribution for the file popularity for different values of the Zipf parameter β.

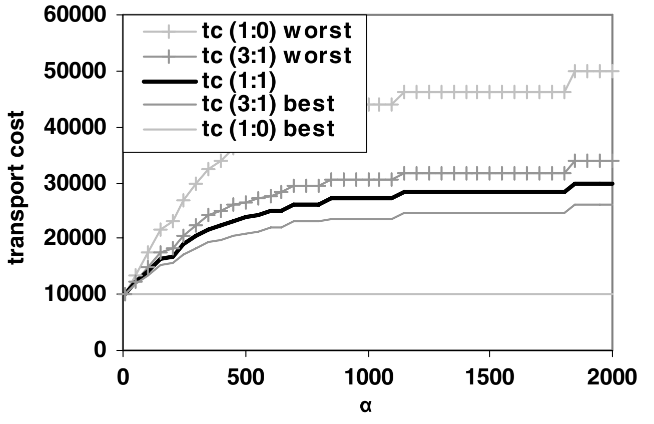


Fig. 7. Transport cost (tc) in a ring network with 8 surrogate servers designed for a symmetric user demand, with an asymmetric user demand (X:Y means that for every X requests at the first surrogate server, Y requests are made at each other surrogate server).

We now look at the increase in transport cost that occurs when this optimally designed network is used for different traffic parameters.

The influence of the Zipf parameter β is rather small. When the actual value of β is 0.5 (50% of all requests are made for the 7 most popular files) or 1.0 (50% requests for top 3) instead of 0.7 (50% requests for top 5), the transport cost is never higher than 2% above the optimal solution (1% higher on average).

The influence of traffic asymmetry is more noticeable. When the storage locations are defined for a symmetrical user demand, the transport cost is given by *tc (1:1)* in Fig. 7. When 3000 requests are made at 1 surrogate server and 1000 requests at each of the other surrogate servers (*tc (3:1)*), the transport costs would normally decrease according to the ILP solution for this asymmetrical design, but in the situation of a symmetrical design the transport cost will depend on the location of the surrogate server with 3000 requests, compared to the storage locations. The actual transport cost will then be somewhere between the best (*tc (3:1) best*) and the worst (*tc (3:1) worst*, 30% higher) case. When all 10,000 requests arrive at 1 surrogate server, the transport cost can be very high (*tc (1:0) worst*) if no optimal (ILP) design is used to take the traffic asymmetry into account.

5. Network design for ring based CDNs with a tree access topology

In a next step, we introduce surrogate servers in the access network and extend the analytical solution presented above. We assume a tree topology consisting of L levels, each with a split x_l (the number of outgoing links for each node at level $l, l = 1 \ldots L$). The links in the access network are unidirectional. Like in the previous section, we study the situation where the request pattern is symmetrical.

5.1. Analytical solution

The least popular objects will be stored on one or more of the N surrogate servers on the ring network (level 0), as described in the previous section. When the number of requests r_i is high enough, storage in the access network becomes beneficial. Due to the symmetry of the problem and the unidirectional access network links (no co-operation possible), an object should be stored at every surrogate server of the appropriate level.

Object o_i will be stored in the lowest level of the access network (level L, closest to the users) when the total cost (cache cost and transmission cost) at that level is lower then the total cost one level higher, or when

$$N \cdot \prod_{j=1}^{L} x_j \cdot \alpha \cdot s_i + r_i \cdot d_i(S_{L,i}) \cdot b_i < N \cdot \prod_{j=1}^{L-1} x_j \cdot \alpha \cdot s_i + r_i \cdot d_i(S_{L-1,i}) \cdot b_i. \tag{36}$$

In general, an object o_i will be stored at level l when

$$N \cdot \prod_{j=1}^{l} x_j \cdot \alpha \cdot (s_{l+1} - 1) > r_i \cdot (d_i(S_{l-1,i}) - d_i(S_{l,i})) > N \cdot \prod_{j=1}^{l-1} x_j \cdot \alpha \cdot (s_l - 1) \tag{37}$$

This means that the algorithm of Fig. 3 has to be modified into the procedure of Fig. 8.

5.2. Experimental results

Using this strategy, the benefits of storage in the access network can be studied on different topologies, similar to the one presented in Fig. 9. A central server is connected to the core ring network, where the edge surrogate servers are located. In the access network, multiple levels of

```
1.   for each $i = 1 .. F$
         Calculate $l$ out of (37)
2.   if $l = 0$:
         Calculate $n_i = \max_{\substack{\Delta C_i(n)>0 \\ 0<n\le N}} n$
     else:
         $n_i = N \cdot \prod_{j=1}^{l} s_j$
3.   optimal surrogate server locations found for all i=1 .. $F$, calculate minimal
     cost from
         $C = \sum_{i=1}^{F} b_i r_i \overline{d_i(n_i)} + \alpha s_i n_i$
```

Fig. 8. Exhaustive strategy to calculate optimal surrogate server location sets for replica placement for a set of objects $O=\{o_1,\ldots,o_F\}$, characterized by request rates r_i, on a ring based CDN with a tree access topology.

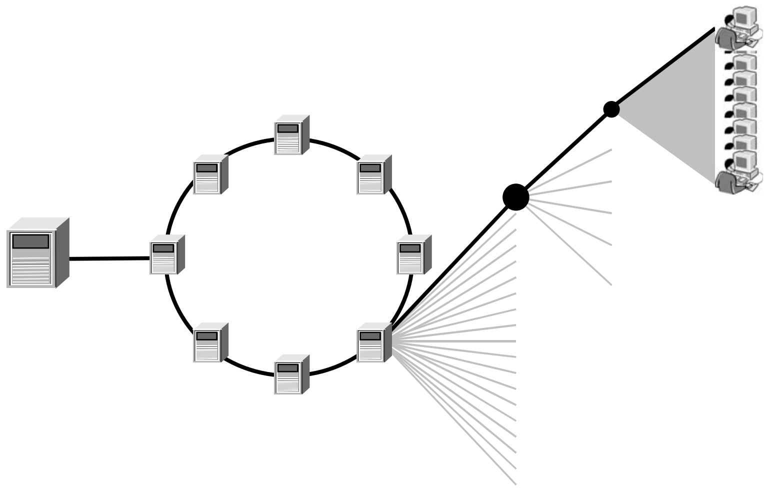


Fig. 9. Ring network with tree access topology (two levels).

aggregation are present, with hub surrogate servers at each level. The users are connected to level one hub surrogate servers, which are in turn grouped together by level two hub surrogate servers in a tree topology.

The influence of the parameter α, the split rate in the access network and the number of edge surrogate servers is shown in Fig. 10, for the analytical solution.

The total number of users and requests is kept constant: 100,000 users and two requests per month per user, for a total number of 500 files. In Fig. 10a the number of edge surrogate servers is constant (4), in Fig. 10b the split rate is constant (4_4 or 4 outgoing links for level one and level two hub surrogate servers) and in Fig. 10c α is constant (0.001). We notice that the efficiency of the hub surrogate servers increases for lower values of α and for denser user populations (lower split rates or less edge surrogate servers, when the number of users is kept constant).

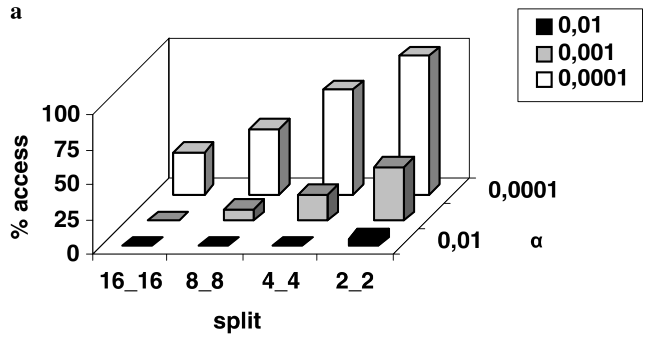
For α = 0.0001, a split rate of 2 per hub surrogate server and four edge surrogate servers (Fig. 10a), all requests are served by the hub surrogate servers in the access network.

6. Dynamic heuristics for content replication

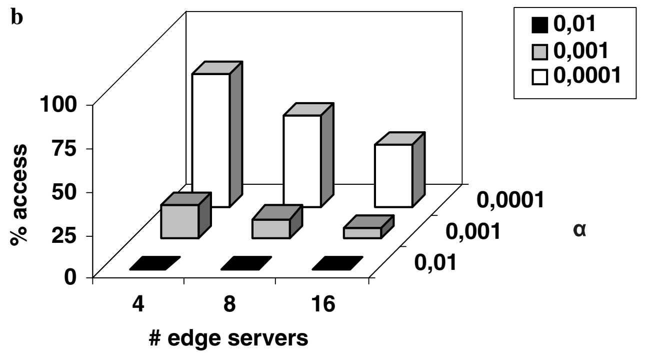
Contrary to the centralized and static solutions in the previous sections, the distributed and dynamic algorithms presented in this section do not calculate global replica placements. Each surrogate server determines by itself, at run-time, which content is stored locally, depending on the traffic passing the node, and dynamically replaces stored content in case of changing request patterns.

Due to the decentralized nature of the algorithms, the results are slightly less optimal than for centralized solutions, but the CDN can now more easily adapt its replica placement to network failures or changes in user behaviour and provide a more robust content distribution service.

a



b



c

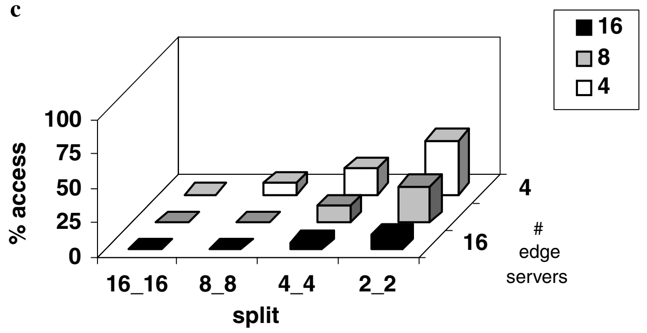


Fig. 10. Influence of the split rate, α and the number of edge surrogate servers on the relative number of requests served by hub surrogate servers in the access network.

6.1. Heuristics

In this section, we present two heuristics based on similar assumptions as for greedy algorithms (based on popularity and distance metrics), with a different point of view on storage costs (limited or unlimited cache sizes). Both algorithms can also introduce specific link costs, which can be used to provide load balancing on the network (see Section 7).

6.1.1. "Survival of the Fittest" heuristic

Every time a file passes one of the surrogate servers, this node will modify a parameter for that file. In this heuristic, this parameter $A_{n,f}$ for file f in node n only depends on the transport cost (amount of bandwidth used):

$$A_{n,f} = T_{n,f}. \tag{38}$$

When a file f passes by node n, the transport cost $T_{n,f}$ is raised by the cost (number of bandwidth units on each link) to transport file f from the source node to node n (this cost would not be required if the file would have been stored in node n). We first store all the passing files until the surrogate server is filled up (limited storage capacity) and then drop stored files in favor of more popular or more distant files (i.e. with higher values for $A_{n,f}$) passing by ("*Survival of the Fittest*", *SF*). Note that this does not necessarily mean that every surrogate server stores the content that is locally most popular. $T_{n,f}$ also depends on the distance to the other nodes storing file f. Therefore it is possible that a very popular file f is not stored in a surrogate server, because another surrogate server nearby already stores a replica of it.

Fig. 11 shows the normalized network and central server load for the core network part (with central server) of Fig. 9. We assume that 500 files are available at the central server. When each of the surrogate servers in the core network can store 100 files, the network load (occupied bandwidth) drops to less than 50% of its maximum value (when no caches are present). The central server load (number of simultaneous streams at the server) even decreases to 30%.

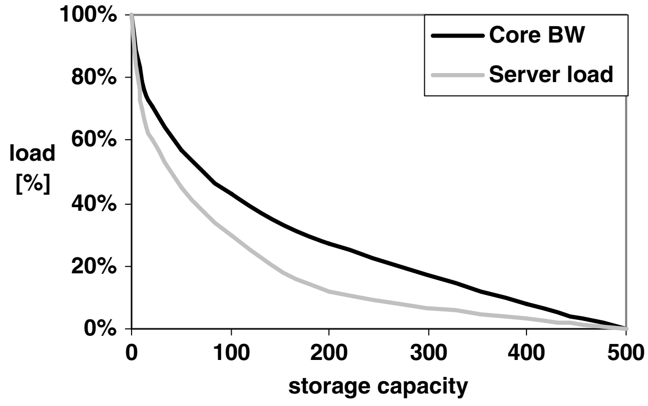


Fig. 11. Network and central server load on a ring network with 8 surrogate servers.

6.1.2. "Storage Renting" heuristic

This algorithm is similar to the first, but now a storage cost is included in the calculation of the parameter $A_{n,f}$ (unlimited cache sizes). When this parameter is positive, the file will be cached (or stay cached), otherwise it will not be cached (or be dropped). The parameter $A_{n,f}$ for file f in node n is calculated as follows:

$$A_{n,f} = T_{n,f} - \alpha \cdot S_{n,f}. \tag{39}$$

Besides the transport cost $T_{n,f}$ a storage cost $S_{n,f}$ is introduced.

$S_{n,f}$ is raised by 1 every time unit file f is stored in node n ("*Storage Renting*", *SR*). This way using a storage slot has a certain cost as well, so that this heuristic can also be used to determine the optimal size of the surrogate servers in the different parts of the network. $S_{n,f}$ is multiplied by the factor α, describing the relative cost between bandwidth and storage. If α is low, only the transport cost is considered to be important, as in the *SF* heuristic. If α is high the storage cost becomes more important and the solution found will have only few stored replicas of the available content.

An example for this heuristic on the topology given in Fig. 9 is shown below. We assume that the central server stores 500 files (e.g. video streams) and that storage slots can be available on the core network as well as on the access network. First all content is only served by the central server, but after a while more files are stored at the surrogate servers (Fig. 12). The storage cost corresponds to the amount of used storage slots (or stored replicas) and is shown as the total cost per level (all level one hub surrogate servers, level two hub surrogate servers or edge surrogate servers).

For the given input parameters, introducing large storage facilities in the access network is not very beneficial: in steady state only 8 files are stored in each level one hub surrogate server, 20 in each level two hub surrogate server and about 180 in each of the edge surrogate servers. When the access network servers would receive more hits (more popular content, a more dense access network, ...)

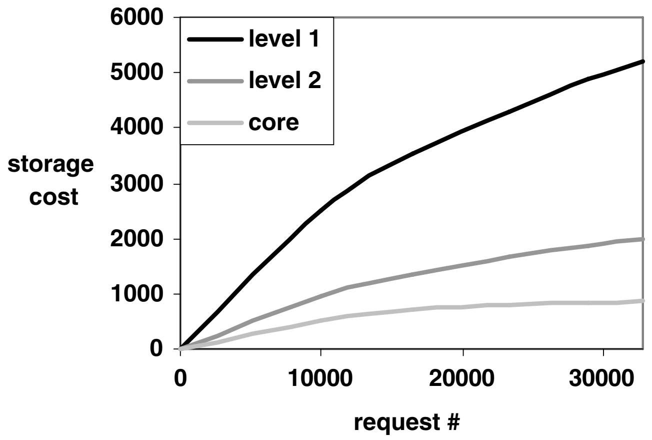


Fig. 12. Storage cost in the core and access network ($\alpha = 0.001$, 500 files, 32,000 user requests, 100 level two hub surrogate servers, 600 level one hub surrogate servers, 100 users per level one hub surrogate server).

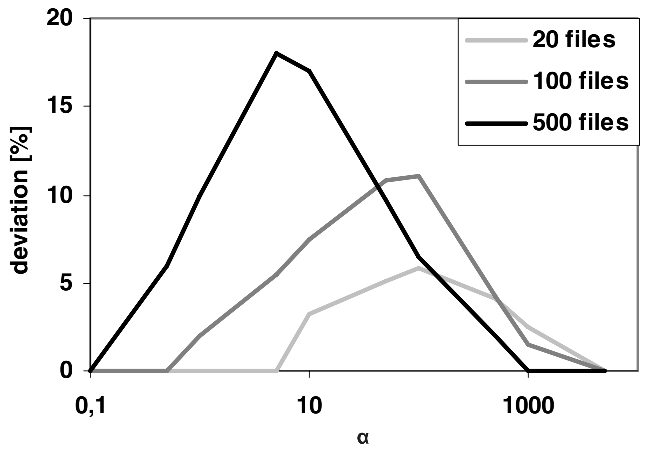


Fig. 13. Deviation from the exact ILP solution (total network cost) for the SR heuristic on a ring network with 8 surrogate servers. Ten thousands requests are made for a variable number of available files.

or when storage is cheaper (lower values for α), storage in the access network could have more advantages.

6.2. Comparison

In this section, the results of the *SR* heuristic are compared to the exact ILP solution for a ring network with 8 surrogate servers and 10,000 requests in total. Fig. 13 shows the average extra network cost (transport costs plus α times the storage cost), caused by the distributed nature of the *SR* heuristic. The results are never worse than 6% above the ILP solution on average (8% for the worst case out of 10 simulations per value of α) for 20 available files, 12% for 100 files (15% worst case) and 18% for 500 files (25% worst case).

Note that the results for the ILP solution in Fig. 13 for a certain value of α correspond to the results for the distributed *SR* heuristic for a value of α that is 10,000 times smaller. This is because the centralized solution calculates the content placement for all 10,000 requests at once, while the distributed solution adapts the content placement after each single request.

7. Dynamic heuristics for content replication with load balancing

7.1. Introduction

To illustrate the importance of load balancing, Fig. 15 shows the bandwidth occupation (in number of simultaneous streams) on the different links of the core network given in Fig. 14 (1 server and 4 surrogate servers).

The *SF* heuristic is used and the surrogate servers can store 100 of all 500 available streams. First the central server serves all requests, but at the steady state situation the surrogate server are filled and serve many requests as well (see also Fig. 4). The outgoing links of the central server (links 1 and 10) are heavily loaded, compared to the other links. Links 2, 4, 7 and 9 are not used at all.

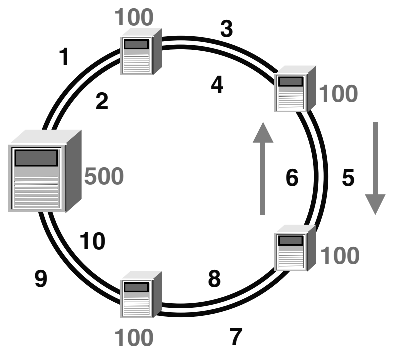


Fig. 14. Core network topology, with uni-directional links.

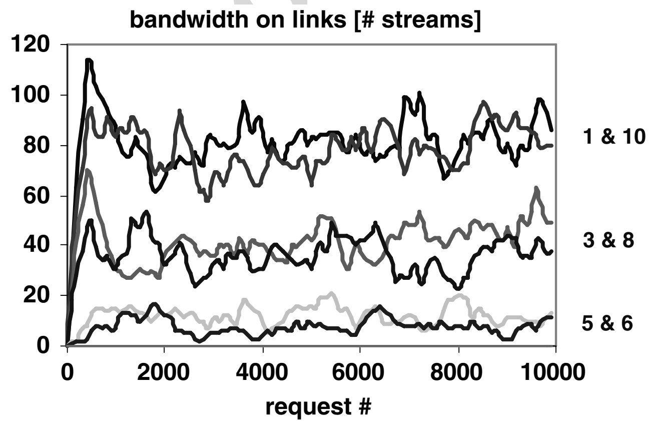


Fig. 15. Bandwidth occupied on the core network links.

When a more uniform load on each of the links could be achieved, a higher number of user requests could be supported by the network. Therefore, the goal of the following heuristics is to minimize the deviation of the actual link loads from the average link load. We assume that the link capacity is uniform on the network.

7.2. Heuristics

Both distributed RPAs can easily be adapted to support load balancing. The only changes in the algorithm are made in the calculation of $T_{n,f}$ as part of the parameter $A_{n,f}$. $T_{n,f}$ represents the streaming cost for file *f* in surrogate server *n*. When a file passes node *n*, $T_{n,f}$ is raised by the cost to transport the file from the source node to node *n*. Until now the cost c_e for using a link was set to 1 for each link. This means that the transport cost between two nodes is proportional to the number of hops between them. Now we change the cost c_e of a link *e* to

$$c_e = \left\lfloor \frac{1}{(1 - l_e)^{\gamma}} \right\rfloor \tag{40}$$

with l_e the actual load on link *e* (in %, relative to the link capacity). Some values for c_e are given in Table 1 (γ is set to 1). When the link load is at 95% of its maximum capacity, the cost for using this link for a new download is 20 times higher than the cost for using a free link. Note

Table 1
Link Cost for a given load

Load	c_e
0	1
0,50	2
0,90	10
0,95	20
0,99	100
0,999	1000

that when the load on the link is smaller than 50% of its capacity, no load balancing is done ($c_e = 1$). By introducing these link costs, the congested links will be avoided when calculating the shortest path (weighted Dijkstra algorithm) between the user and the candidate surrogate servers storing the requested file. Even when a congested link has to be used, the values for $T_{n,f}$ (and consequently $A_{n,f}$) will be higher for all nodes n after the congested link(s) on the path. Therefore more content will be stored beyond the congested link(s).

The situation in Fig. 15 now changes to that in Fig. 16. The load on all links is now much closer to the average value (the variance is much lower) and links 4 and 7 also carry streams. Note that the average value of the total link load will be higher in the load balanced situation, compared to the original case, where the total bandwidth occupation on the network was minimized. Spreading the load over all the network links will therefore also slightly increase the average load.

7.3. Experimental results

To study these extended heuristics, simulations were performed on a network with a central server connected to a core ring with 8 surrogate servers (like on Fig. 9). On average 450 streams are present on the network, 500 files (with Zipf-like popularity distribution) are available. The requests (10,000 in total) are uniformly distributed over the different destination nodes and served over the least congested path. The results for the *SF* heuristic are

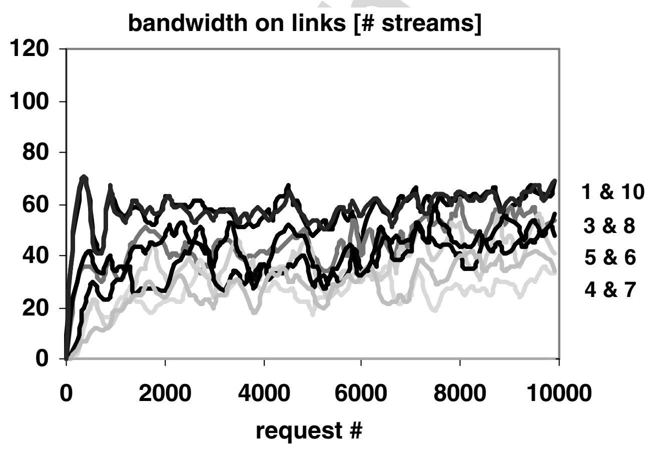


Fig. 16. Bandwidth occupied on the load balanced core network links.

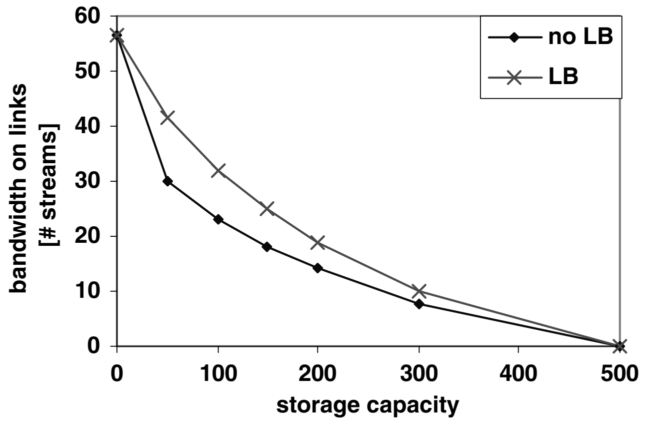


Fig. 17. Average bandwidth on the core network links, with (LB) and without (no LB) load balancing.

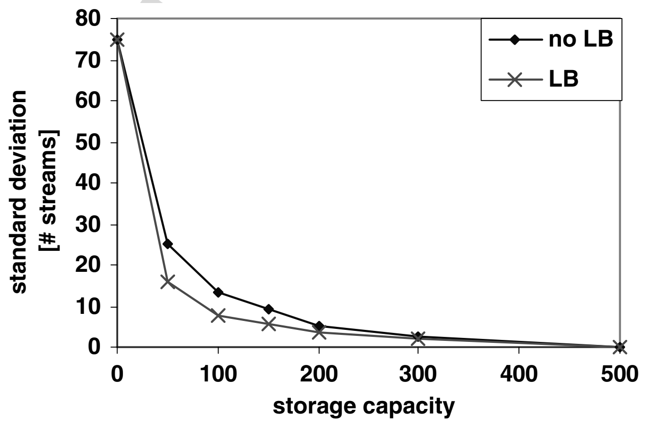


Fig. 18. Standard deviation on the bandwidth on the core network links, with (LB) and without (no LB) load balancing.

compared for different situations: with or without load balancing and for different cache sizes. The parameter γ is set to one.

In Fig. 17 the influence of load balancing on the average bandwidth on the core network links is shown. For intermediate cache sizes on the surrogate servers, the average bandwidth is up to 40% higher than in the optimal situation without load balancing. However, the deviation of the actual link load around this average is much lower, as shown in Fig. 18.

Balancing the load on the network comes at the price of a higher average link bandwidth. The influence of the parameter γ is not clearly visible on this network topology, since the content placement is already near the optimum. Simulations on more complex topologies fall outside of the scope of this paper, but show that larger values for γ increase the level of load balancing (higher average bandwidth and even lower values for the standard deviation) [17].

8. Conclusion

In the first part of this paper, we have presented a static solution for the replica placement problem in

CDNs. We have compared an ILP formulation of the problem to the analytical solution for ring based CDNs. Both approaches showed that the load on the network and the central server can be considerably decreased. Adding storage facilities in the access network or even at the home network can be cost effective and might be interesting to study in future work.

Afterwards two distributed algorithms have been introduced. These heuristics dynamically adapt the replica placement to variations in the network load, user behaviour or available content. This way congestion in the network can be avoided and a more robust service can be provided, at the price of a slightly increased network load.

The introduction of link costs in the load balancing algorithms can also be used for different objectives. Instead of specifying the link load, they could also represent transfer or propagation delays. Minimizing link delays together with other network resources can be interesting for future work. While the approach of storing whole files, as presented in this paper, is very effective for Video on Demand services, a method of storing partial files (e.g. with sliding intervals) can be interesting for very popular content (e.g. live television) and will also be studied in future work.

References

[1] Akamai, <http://www.akamai.com>.

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web caching and Zipf-like distributions: evidence and implications, IEEE Infocom. (1999).

[3] P. Backx, T. Wauters, B. Dhoedt, P. Demeester, A comparison of peer-to-peer architectures, Eurescom. Summit. (2002).

[4] L. Qiu, V.N. Padmanabhan, G.M. Voelker, On the placement of web server replicas, IEEE Infocom. (2001).

[5] M. Karlsson, M. Mahalingam, Do We Need Replica Placement Algorithms in Content Delivery Networks?", Seventh International Web Content Caching and Distribution Workshop, August 2002.

[6] M. Karlsson, C. Karamanolis, M. Mahalingam, A Framework for Evaluating Replica Placement Algorithms, Technical Report HPL-2002, HP Laboratories, July 2002.

[7] J. Kangasharju, J. Roberts, K. Ross, Object replication strategies in content distribution networks, Comput. Commun. 25 (4) (2002) 376–383.

[8] D. Turrini, F. Panzieri, Using P2P techniques for content distribution internetworking: a research proposal, in: Second International Conference on Peer-to-Peer Computing, 2002.

[9] T. Wauters, J. Coppens, T. Lambrecht, B. Dhoedt, P. Demeester, Distributed replica algorithms for peer-to-peer content distribution networks, in: EuroMicro Conference, September 2003.

[10] J.M. Almeida, D.L. Eager, M.K. Vernon, A hybrid caching strategy for streaming media files, in: Proceedings of Multimedia Computing and Networking (MMCN), San Jose, CA, January 2001.

[11] X. Tang, J. Xu, On replica placement for QoS-aware content distribution, in: IEEE Infocom – The Conference on Computer Communications, March 2004.

[12] M. Yang, Z. Fei, A model for replica placement in content distribution networks for multimedia applications, in: ICC 2003 – IEEE International Conference on Communications, May 2003.

[13] M. Karlsson, M. Mahalingam, Choosing replica placement heuristics for wide-area systems, in: Proceedings of the International Conference on Distributed Computing Systems (ICDCS), March 2004.

[14] I. Cidon, S. Kutten, R. Soffer, Optimal allocation of electronic content, in: Proceedings of IEEE Infocom, April 2001.

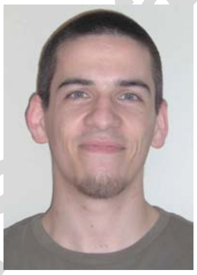
[15] J. Coppens, T. Wauters, F. De Turck, B. Dhoedt, P. Demeester, Evaluation of a monitoring based architecture for delivery of high quality multimedia content, in: Conference Proceedings of Tenth IEEE Symposium on Computers and Communications ISCC 2005, Cartagena, Spain, June 27–30, 2005.

[16] J. Coppens, T. Wauters, F. De Turck, B. Dhoedt, P. Demeester, Evaluation of replica placement and retrieval algorithms in self-organizing CDNs, in: Conference Proceedings of IFIP/IEEE International Workshop on Self-Managed Systems & Services SelfMan 2005, Nice, France, May 19, 2005.

[17] T. Wauters, J. Coppens, B. Dhoedt, P. Demeester, Load balancing through efficient distributed content placement, in: Conference Proceedings of NGI 2005, Rome, Italy, April 18–20, 2005.

[18] J. Liu, J. Xu, Proxy caching techniques for media streaming over the Internet, IEEE Commun. Mag. 42 (8) (2004) 88–94.

[19] Cplex, <http://www.ilog.com/products/cplex/>.



Tim Wauters received his M.Sc. degree in electrotechnical engineering (option communication techniques) in 2001 from the University of Ghent, Belgium. Since September 2001, he has been working on the design of content distribution and peer-to-peer networks in the Department of Information Technology (INTEC), at the same university. His work has been published in the proceedings of several international conferences and journals.

Jan Coppens joined the IBCN research group in 2001 after receiving his M.Sc. degree in computer science at Ghent University, Belgium. He has been active in several IST projects (Tequila, Scampi) and his main research interests are Quality of Service, traffic engineering, network monitoring and distributed computing. His current work focusses on adaptive and self-organizing content distribution networks. Jan Coppens is author or co-author of several scientific publications in the proceedings of international conferences and journals.

Filip De Turck received his M.Sc. degree in Electronic Engineering from the Ghent University, Belgium, in June 1997. In May 2002, he obtained the Ph.D. degree in Electronic Engineering from the same university. At the moment, he is a part-time professor and a post-doctoral fellow of the F.W.O.-V., affiliated with the Department of Information Technology of the Ghent University. Filip De Turck is author or co-author of approximately 80 papers published in international journals or in the proceedings of international conferences. His main research interests include scalable software architectures for telecommunication network and service management, performance evaluation and optimization of routing, admission control and traffic management in telecommunication systems.

Bart Dhoedt received a degree in Engineering from the Ghent University in 1990. His research, addressing the use of micro-optics to realize parallel free space optical interconnects, resulted in a Ph.D. degree in 1995, at the Department of Information Technology of the Faculty of Applied Sciences, University of Ghent. After a 2 year post-doc in opto-electronics, he became professor at the same university. Since then, he is responsible for several courses on algorithms, programming and software development. Bart Dhoedt is author or co-author of approximately 100 papers published in international journals or in the proceedings of international conferences. His current research addresses software technologies for communication networks, peer-to-peer networks, mobile networks and active networks.

Piet Demeester received the Masters degree in Electro-technical engineering and the Ph.D. degree from the Ghent University, Gent, Belgium in 1984 and 1988, respectively. In 1992 he started a new research activity on broadband communication networks resulting in the IBCN-group (INTEC Broadband communications network research group). Since 1993 he became professor at the Ghent University where he is responsible for the research and education on communication networks. The research activities cover various communication networks (IP, ATM, SDH, WDM, access, active, and mobile), including network planning, network and service management, telecom software, internetworking, network protocols for QoS support, etc. Piet Demeester is author of more than 400 publications and member of the editorial board of several in ternational journals and technical program committees (ECOC, OFC, DRCN, ICCCN, IZS, and &).