Replica Placement in Adaptive Content Distribution Networks

Sven Buchholz
Dresden University of Technology
Department of Computer Science
D-01062 Dresden, Germany
buchholz@rn.inf.tu-dresden.de

Thomas Buchholz
Ludwig-Maximilians-University
Department of Computer Science
D-80358 Munich, Germany
buchholz@informatik.uni-muenchen.de

ABSTRACT

Adaptive content networking is a promising new approach aimed at scalable delivery of content to a pervasive client population. By adaptive content delivery networks (A-CDN) content is adapted, replicated and delivered to the clients in a cost-quality-optimized fashion. The integration of content adaptation into CDNs minimizes the interference of adaptation with replication effectiveness.

The paper presents ongoing research on replica placement in A-CDNs. Based on a static model for cost-quality-optimized adaptive content networking, algorithms to optimize the placement of replicas in the surrogates of an A-CDN are discussed. The dynamics of a real Web scenario are not explicitly taken into account by the algorithms. Whereas long-term dynamics are dealt with by periodic adjustments of the underlying model and recalculation of an optimal placement, short term dynamics are considered to result in inaccuracies in the system and load model. Therefore, algorithms being tolerant to an imperfect underlying model are chosen.

As adaptation path composition turns out to be a sub-problem of replica placement in A-CDNs, we also introduce an algorithm for optimal adaptation path composition.

Keywords

CDN, content adaptation, replica placement, adaptation path composition

1. INTRODUCTION

In the upcoming world of Pervasive Computing, users access information in the Internet by a huge variety of heterogeneous devices. The devices are attached to the Internet by various communication systems featuring heterogeneous characteristics. The key to cope with the heterogeneity is the adaptation of the representation of content in order to meet the media handling capabilities of the devices and the transmission restrictions imposed by the networks. In previous research, a lot of effort has been spent in the field of content adaptation. Nonetheless, previous approaches have been designed without taking the capabilities of Content Distribution Networks into account. Content Distribution Networks are applied in todays World Wide Web to improve performance and scalability of content delivery by replicating content on distributed nodes, so-called surrogates. However, content adaptation interferes with the effectiveness of replication. This issue has been examined in detail in [3].

Leveraging the advantages of Content Distribution Networks in Pervasive Computing environments where content adaptation is necessary to meet the heterogeneity of the clients is subject of our research. We envision an Adaptive Content Distribution Network (A-CDN) that may flexibly replicate different representations of objects in its surrogates and apply content adaptation within the CDN to finally adapt and deliver content to the clients. We allow clients to accept different representations of each object having different qualities w.r.t. the media handling capabilities of the particular client. The selection of an appropriate representation to satisfy a client's request is done in a cost-quality-optimized manner.

This paper deals with the problem of replica placement (RP) in A-CDNs. RP in A-CDNs deals with the question which representation of which object to store in which surrogate of the A-CDN in order to satisfy all requests globally optimal w.r.t. cost and quality. Therefore, the RP algorithm is required to be aware of the adaptation capabilities of the A-CDN. We present ongoing research on adaptation-aware RP algorithms. They are based on a static model for cost-quality-optimized adaptive content networking. Hence, the algorithms perform static optimization. In order to capture dynamic changes in the underlying system and in the load of requests, the placement of replicas must be recalculated regularly. The recalculations are always based on an readjusted system and load model whereby system state and load are predicted based on averaged usage statistics.

The remainder of the paper is organized as follows. In the next section, we present our model for cost-quality-optimized adaptive content networking and formalize the objective in cost-quality-optimized adaptive content networking. Section 3 presents the algorithms for adaptation-aware RP in A-CDNs. As the composition of cost-quality-optimized adaptation paths turns out to be a sub-problem of the RP problem, Section 3 also comprises a solution to this problem. Related work is discussed in Section 4. Finally, concluding remarks are given in Section 5.

SAC'04 March 14-17, 2004, Nicosia, Cyprus

2. MODEL FOR COST-QUALITY-OPTIMIZED CONTENT NETWORKING

This section describes the model for cost-quality-optimized adaptive content networking. At first, the system model is introduced. It captures the physical components involved in the delivery of contents, their properties, and capabilities.

The set of nodes engaged in content delivery is N. It comprises the origin servers $b \in B$, hosting the primary copies of the objects, the surrogates of the CDN $h \in H$, and client nodes $c \in C$. In order to mask the mobility of individual clients and to reduce complexity, a client node $c \in C$ does not represent an individual physical client. In contrast, we join all physical clients sharing the same hardware and software configuration as well as the same network connectivity to one client node c that produces the aggregated load of the joint physical clients. Thereby the mobility of clients maps to load changes (for load modeling see below).

K denotes the set of objects that is distributed by the CDN. Each object $k \in K$ can be delivered in different representations r from the set R of representations. A representation is characterized by a collection of media features according to [14]. The capabilities of a client to handle objects in a certain representation, the media handling capabilities, are specified by a predicate over the media features as defined by IETF Media Feature Sets [14]. Formally, the media handling capabilities of a client represented by c define a set $mhc(c) \subseteq R$ of representations that can be handled by the client. The quality of a representation $r \in R$ of an object $k \in K$ experienced by a client represented by $c \in C$ is captured by the quality function $qf\colon C \times K \times R \to \mathbb{R}[0,1]$. A quality of 1 refers to the maximum achievable quality. Zero quality means r of k is useless to c. Quality function we assume to be designed according to the concept of multidimensional quality functions as described in [2].

A tuple $(k,n,r) \in K \times N \times R$ specifies an instance of object k in representation r within node n. However, the term instance does not necessarily imply that k is materialized as a replica in n. The placement of primary replicas on origin servers is determined by the relation $P_0 \subseteq K \times B \times R$, viz. an instance $(k,b,r) \in P_0$ means origin server b stores object k in representation r. Analogously, $P_\mathrm{R} \subseteq K \times H \times R$ is the placement of replicas in the surrogates of the CDN. It is constrained by the restricted storage capacity $sc(h)$ of each surrogate h:

$$\forall h \in H \ \sum\nolimits_{(k,h,r)\in P_\mathrm{R}} sf(k,r) \leq sc(h), \qquad (1)$$

where $sf(k,r)$ denotes the size of object k in representation r. Formally, the function $sc\colon N \to \mathbb{N}$ of storage capacities assigns 0 to all non-surrogate nodes and the capacity in bytes as a positive integer to the surrogates $h \in H$.

A set O comprises operations $o\colon K \times N \times R \to N \times R$ describing both (1) adaptation operations $o_\mathrm{adapt}\colon (k,h,r) \mapsto (h,r')$, performed at surrogate h to alter the representation of object k from r to r', and (2) transfer operations $o_\mathrm{trans}\colon (k,n,r) \mapsto (n',r)$[1], each representing a unidirectional network link from n to n'. Bidirectional links are represented by one operation for either direction. Operations are assumed to be uncapacitated but to incur costs. The cost of applying an operation o to an instance (k,n,r) of object k is captured by the cost function $cf_o\colon K \times N \times R \to \mathbb{R}$. The vector (cf_o) comprises the cost functions of all operations $o \in O$.

To deliver objects operations are chained to form adaptation paths. We model an adaptation path as a directed unary tree $\pi = \langle (k,n_0,r_0) \xrightarrow{o_1} (k,n_1,r_1) \xrightarrow{o_2} \cdots \xrightarrow{o_p} (k,n_p,r_p) \rangle$. An edge $e_i \in E(\pi)$ represents the application of an operation o_i in π. Apart from the root, which is a replica $(k,n_0,r_0) \in (P_0 \cup P_\mathrm{R})$, the vertices (k,n_i,r_i) are the instances yielded by applying o_i on the respective predecessor (k,n_{i-1},r_{i-1}). The cost $cost_\pi$ of an adaptation path π is the sum of the costs associated with the operations in the path. Its quality $qual_{\pi,c}$ is the quality $qf(k,n_p,r_p)$ of the leaf vertex.

According to the above definitions, we formally define a system for adaptive content delivery as a 10-tuple $\Sigma = \big(K, R, sf, N, sc, P_0, mhc, qf, O, (cf_o)\big)$ (Note: The subsets B, H, and C of N are implicitly defined by P_0, sc, and mhc, respectively, and need not be included in the tuple Σ.).

The requests of the CDN's clients are captured by the load model. A pair $(c,k) \in C \times K$ specifies a request of a client represented by client node c for object k. A request (c,k) can be satisfied by constructing an adaptation path π delivering the object k to the client c in a representation r that can be handled by the client ($r \in mhc(c)$) and has non-zero quality ($qf(c,k,r) > 0$). The matrix $\Lambda = (\lambda_{c,k})$ of request rates of all requests $(c,k) \in C \times K$ describes the load of the system Σ. It is estimated based on usage statistics.

There are two conflicting goals in cost-quality-optimized adaptive content networking: maximizing quality and minimizing cost. We tackle this problem by projecting quality into the cost domain. This is accomplished by assigning revenue to quality. We assume the customers' willingness to pay to be proportional to the experienced quality. Consequently, the revenue yielded by satisfying a request (c,k) by the adaptation path π is $rev_{\pi,c} = price_k \cdot qual_{\pi,c}$. The proportionality factor $price_k$ is the nominal price of k, i.e. the amount the customer is willing to pay for object k if it is delivered with optimal quality. Thus, the profit yielded by satisfying a single request (c,k) by the adaptation path π is $profit_c(\pi) = price_k \cdot qual_{\pi,c} - cost_\pi$. The objective in cost-quality-optimized adaptive content networking is maximizing the overall profit

$$Profit_\Lambda(P_\mathrm{R}) = \sum\nolimits_{(c,k)\in C\times K} \lambda_{c,k} \cdot profit_{c,\pi_{c,k}^\mathrm{opt}} \qquad (2)$$

yielded by responding to every request (c,k) by the optimal adaptation path $\pi_{c,k}^\mathrm{opt}$ satisfying (c,k) w.r.t. a placement P_R. The optimization problem is subject to the constrained storage capacity (Eq. 1).

3. ADAPTATION-AWARE REPLICA PLACEMENT IN A-CDNS

Replica placement in A-CDNs is targeted at finding a placement P_R yielding maximum overall profit $Profit_\Lambda(P_\mathrm{R})$ (Eq. 2) subject to the storage capacity constraint (Eq. 1). Determining overall profit $Profit_\Lambda(P_\mathrm{R})$ requires the profit of the optimal adaptation path $\pi_{c,k}^\mathrm{opt}$ for each request (c,k) with $\lambda_{c,k} > 0$ to be known. Hence, composing optimal adaptation paths subject to a given P_R is a sub-problem of the RP problem. It is discussed in Section 3.1. The actual

[1]In the paper, we assume transfer operations not to affect the media features of objects (hence r maps to r). Nonetheless, the comprehensive definition of operations allows for transfer operations $(k,n,r) \mapsto (n',r')$ affecting the media features, too (e.g. unreliable streaming transfer resulting in loss in signal-to-noise ratio).

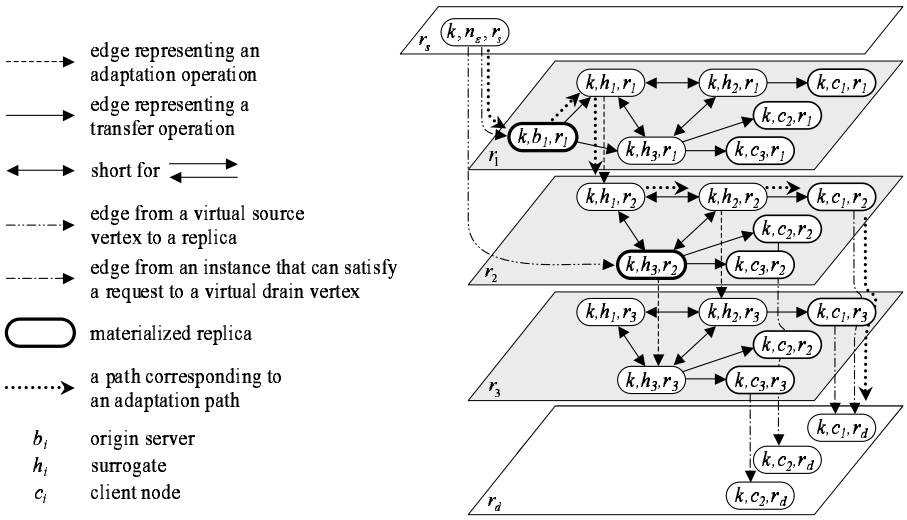

Figure 1: Example of a connected component Γ_k of a virtual system graph Γ

RP algorithm using the approach of Section 3.1 to calculate $Profit_{\Lambda}(P_{\mathrm{R}})$ is dealt with in Section 3.2.

3.1 Adaptation Path Composition

Our solution to this sub-problem is based on the general approach of mapping adaptation path composition onto conventional shortest path search in weighted graphs as introduced by [5]. However, we extend the ideas of [5] to allow for cost-quality-optimized adaptation paths and for multiple different replicas of an object as potential root vertices of the adaptation path.

The approach requires the system model Σ to be mapped onto a weighted directed graph Γ. We call Γ the virtual system graph. The set $V(\Gamma)$ of vertices in Γ comprises all instances $(k, n, r) \in K \times N \times R$ that can potentially be vertices in an adaptation path in Σ. The instances are connected by edges $e \in E(\Gamma)$ representing the application of adaptation or transfer operations. Every $((k, n, r), (k', n', r')) \in (K \times N \times R) \times (K \times N \times R)$ where $k = k'$ and there is an operation $o \in O$ so that $(n', r') = o(k, n, r)$ is an edge in Γ.

Provided that the underlying network is not partitioned and all replicas can be obtained from one master copy by a sequence of adaptation and transfer operations, the resulting virtual system graph Γ is a disconnected graph with a connected component Γ_k for every object $k \in K$. Otherwise their might be multiple components per object. As their are no operations to convert an object into another one, there are no edges connecting the components of different objects.

Each component can be viewed as having multiple horizontal layers where all vertices within the same layer stand for the same representation r_i of object k within different nodes. Vertices in different layers that lie one upon the other stand for different representations of k within the same physical node n_i. One component of a sample virtual system graph is shown in Figure 1 (please note: Figure 1 contains an additional r_s- and r_d-layer, that will be explained below).

Every path in Γ that originates in an instance that is materialized as a replica $(k, n_0, r_0) \in (P_0 \cup P_{\mathrm{R}})$ and ends in an instance (k, c, r_p) that can satisfy a request (c, k) (i.e. $r_p \in mhc(c)$ and $qf(c, k, r_p) > 0$) corresponds to an adaptation path π in Σ. In order to map the search for an optimal adaptation path onto a single-source single-destination shortest path search in Γ we introduce two additional sets of vertices of Γ: virtual source and virtual drain vertices.

There is a virtual source vertex (k, n_ε, r_s) for each $k \in K$. It is added as the source vertex to all paths in Γ that correspond to an adaptation path for k. Each virtual source vertex (k, n_ε, r_s) is connected by an outgoing edge to every replica of k. Moreover, there is a virtual drain vertex (k, c, r_d) for every request $(c, k) \in C \times K$ having an ingoing edge from every instance that can satisfy the request (c, k). A virtual drain vertex (k, c, r_d) is added as the destination to all paths in Γ that correspond to an adaptation path that can satisfy a request (c, k). The virtual source vertex of the sample component Γ_k in Figure 1 is depicted in the top layer. Virtual drain vertices are shown in the bottom layer.

By an adequate definition of the weight function $w : E(\Gamma) \to \mathbb{R}$, the search for the optimal adaptation path $\pi^{\mathrm{opt}}_{c,k}$ that can satisfy a request (c, k) maps to the search for the shortest path between the virtual source vertex (k, n_ε, r_s) and the virtual drain vertex (k, c, r_d). An appropriate weight function is given in the following. The weight of an edge $e = ((k, n, r), (k, n', r'))$ that stands for the application of an adaptation operation is the cost of the operation plus the loss in revenue due to quality degradation: $w(e) = cf_o(k, n, r) + price_k \cdot (qf(c, k, r) - qf(c, k, r'))$. Edges pointing from a virtual source vertex (k, n_ε, r_s) to a replica $(k, n, r) \in (P_0 \cup P_{\mathrm{R}})$ are weighted by the negated revenue of the initial representation of the replica: $w(e) = -price_k \cdot qf(c, k, r)$. Thereby the different qualities of different replicas are incorporated into the lengths of the paths in Γ. The edges destined to virtual drain vertices are neutral w.r.t. the path length. Accordingly they have zero weights: $w(e) = 0$.

Thus, the length of a path between a virtual source and a virtual drain in Γ, which corresponds to an adaptation path π, is $-profit_c(\pi)$; i.e. minimizing the path length results in maximized $profit_c(\pi)$ of the adaptation path π. Hence, an optimal adaptation path $\pi^{\mathrm{opt}}_{c,k}$ can be determined using conventional shortest path algorithms such

as Bellman-Ford or Dijkstra[2][6]. The search can be restricted to the component Γ_k. Furthermore, client nodes other than the requesting client node are never intermediary nodes within the adaptation path. Accordingly, all vertices $(k', c', r') \in (K \times C \times R)$ where $c' \neq c$ can be neglected in the search for $\pi^{\text{opt}}_{c,k}$. Thus, the computational complexity of the adaptation path composition has a worst case upper bound of $O\big(|H \cup B|^2 |R|^2\big) = O\big(|H|^2 |B|^2 |R|^2\big)$ with Dijkstra's shortest path algorithm and $O\big(|H \cup B|^3 |R|^3\big) = O\big(|H|^3 |B|^3 |R|^3\big)$ with the Bellman-Ford algorithm.

3.2 Replica Placement Algorithms

Finding an optimal placement P_{R} subject to the storage capacity constraint is NP-hard. There is a reduction to the knapsack optimization problem. The proof is based on the finding that the general RP problem, i.e. without content adaptation within the CDN, is a special case of our problem. A reduction of the general RP problem to the knapsack problem is given by [12].

Due to NP-hardness, finding an optimal solution is infeasible. Instead we favor heuristics to tackle the problem. Plain and greedy ranking heuristics have proven feasibility for general RP [12, 13, 15] as well as for the view selection problem [1, 9], dealing with the selection of aggregated views to materialize in data warehouses. Those problems are similar to the adaptation-aware RP problem in A-CDNs. Accordingly, we consider ranking algorithms promising for our problem.

Ranking heuristics [13] construct a solution by (1) identifying a set of partial solutions that can be combined to construct the overall solution, (2) ranking the alternative partial solutions according to a benefit function, and (3) selecting a subset of the partial solution in order of their rank and combine them to the overall solution while skipping partial solutions that violate a constraint.

In the context of adaptation-aware RP, an overall solution is a set P_{R} of replicas. It can be decomposed into partial solution being the materialization of a single replica (k_i, h_i, r_i). The overall solution is the union of the selected replicas: $P_{\text{R}} = \bigcup_i \{(k_i, h_i, r_i)\}$. The selection of replicas is checked against the size constraints of the surrogates (Eq. 1).

In order to maximize the overall profit, the ranking and the selection of replicas is based on the profit yielded by materializing the particular replicas. However, we do not have defined the profit of single replicas but only the profit $Profit_\Lambda(P_{\text{R}})$ of an entire placement P_{R} of replicas. Hence, we need to calculate the profit of a single replica (k, h, r) as the gain $Profit_\Lambda(\{(k, h, r)\} \cup \overline{P_{\text{R}}}) - Profit_\Lambda(\overline{P_{\text{R}}})$ in profit yielded by materializing (k, h, r) in addition to an already materialized reference placement $\overline{P_{\text{R}}}$. To allow for the different consumption of the constrained storage capacity by the different replicas, the gain in profit must be put in proportion to the replica's size $sf(k, r)$. Hence, we obtain the benefit function:

$$benefit_\Lambda((k, h, r), \overline{P_{\text{R}}}) = \frac{Profit_\Lambda(\{(k, h, r)\} \cup \overline{P_{\text{R}}}) - Profit_\Lambda(\overline{P_{\text{R}}})}{sf(k, r)} \quad (3)$$

With the plain ranking heuristic, the potential replicas (k_i, h_i, r_i) are ranked once according to their benefit $benefit_\Lambda((k_i, h_i, r_i), \emptyset)$ w.r.t. an empty reference placement $\emptyset$ (i.e. only primary replicas in origin servers). The overall solution is constructed based on this initial ranking (Fig. 2). As opposed to that, greedy ranking means the benefits are recomputed after each step of selecting a replica. The reference placements $\overline{P_{\text{R}}}$ in subsequent steps are the placements determined in the respective previous steps, while the first step assumes $\overline{P_{\text{R}}} = \emptyset$ (Fig. 3).

By means of greedy ranking, we expect potentially better results than with plain ranking whereas the goodness of a placement P_{R} corresponds to $Profit_\Lambda(P_{\text{R}})$. The gain in goodness of the placement produced by the greedy ranking algorithm comes from the fact that the benefit of a replica significantly depends on the placement of other replicas of the same object. Nevertheless, it is paired with the penalty of higher computational complexity. The worst case upper

```
▷ computation of the benefit functions
  and ranking of the partial solutions
Ranking := {};                                  initializing ranking list
foreach k ∈ K
    p_0 := 0;                                   profit yielded with empty placement ∅
                                                by responding to all requests for k
    foreach c ∈ C
        compute π^opt_{c,k} w.r.t. empty placement ∅;
        p_0 := p_0 + λ_{c,k} · profit_{c,π^opt_{c,k}};
    foreach (h, r) ∈ H × R
        p := 0;                                 profit yielded with placement {(k, h, r)}
                                                by responding to all requests for k
        foreach c ∈ C
            compute π^opt_{c,k} w.r.t. placement {(k, h, r)};
            p := p + λ_{c,k} · profit_{c,π^opt_{c,k}};
        b := (p − p_0)/sf(k, r);                benefit of (k, h, r)
        add ({(k, h, r)}, b) to list Ranking sorted by b
        in descending order;
▷ construing the overall solution by
  selecting partial solutions
P_R := ∅;                                       initializing placement
OSC := 0;
foreach h ∈ H                                   initializing
    SC[h] := sc(h);                             array of remaining capacities
    OSC := OSC + sc(h);                         remaining overall capacity
while (not empty(Ranking)) and (not OSC = 0)
    ({(k, h, r)}, b) := head(Ranking);
    Ranking := tail(Ranking);
    if SC[h] > sf(k, r) then                    checking size constraint
        P_R := P_R ∪ {(k, h, r)};               adding partial solution
        SC[h] := SC[h] − sf(k, r);
        OSC := OSC − sf(k, r);
return P_R;
```

Figure 2: Plain ranking algorithm

[2]The Dijkstra algorithm works with non-negative weights only. In case of the weight function w, the edges originating in the r_s-layer have negative weights. Nonetheless, if the weights of all other edges are non-negative, an extension of the Dijkstra algorithm (viz. marking the destinations (k, n, r) of the edges $e = ((k, n_\varepsilon, r_s), (k, n, r))$ as initially permanent with a distance of $w(e) = -price_k \cdot qf(c, k, r)$) works and is preferable to Bellman-Ford thanks to lower complexity.

bound of plain ranking is

$$O\big(|C||K||H|^3|B|^2|R|^3 \log|K|\big) \quad (4)$$

while greedy ranking has an upper bound of

$$O\big(|C||K||H|^4|B|^2|R|^4 \log|K|\big). \quad (5)$$

```
▷ initial computation of the benefit functions
   and ranking of the partial solutions
Ranking := {};                                  initializing ranking list
foreach k ∈ K
   p_0 := 0;                          profit yielded with empty placement ∅
                                        by responding to all requests for k
   foreach c ∈ C
      compute π^opt_{c,k} w.r.t. empty placement ∅;
      p_0 := p_0 + λ_{c,k} · profit_{c,π^opt_{c,k}};
   foreach (h, r) ∈ H × R
      p := 0;                      profit yielded with placement {(k, h, r)}
                                        by responding to all requests for k
      foreach c ∈ C
         compute π^opt_{c,k} w.r.t. placement {(k, h, r)};
         p := p + λ_{c,k} · profit_{c,π^opt_{c,k}};
      b := (p − p_0)/sf(k, r);                  benefit of (k, h, r)
      add ({(k, h, r)}, b, p) to list Ranking sorted by b
      in descending order;
▷ construing the overall solution by selecting partial
   solutions and recomputing the benefits after each step
P_R := ∅;                                       initializing placement
OSC := 0;
foreach h ∈ H                                                 initializing
   SC[h] := sc(h);                              array of remaining capacities
   OSC := OSC + sc(h);                          remaining overall capacity
while (not empty(Ranking)) and (not OSC = 0)
   ({(k, h, r)}, b, p_0) := head(Ranking);
            note: p_0 is set to the profit yielded with placement
             {(k, h, r)} ∪ P_R by responding to all requests for k
   Ranking := tail(Ranking);
   if SC[h] > sf(k, r) then                     checking size constraint
      P_R := P_R ∪ {(k, h, r)};                 adding partial solution
      SC[h] := SC[h] − sf(k, r);
      OSC := OSC − sf(k, r);
      ▷ removing replicas of k from ranking
      foreach ({(k', h', r')}, b', p') ∈ Ranking
         if k' = k then
            remove ({(k', h', r')}, b', p') from Ranking;
      ▷ recomputing the benefits of replicas of k
        and reinserting them into the ranking
      foreach (h, r) ∈ H × R
         p := 0;                        profit yielded with placement
                                          {(k, h, r)} ∪ P_R by responding
                                                        to all requests for k
         foreach c ∈ C
            compute π^opt_{c,k} w.r.t. {(k, h, r)} ∪ P_R;
            p := p + λ_{c,k} · profit_{c,π^opt_{c,k}};
         b := (p − p_0)/sf(k, r);                 benefit of (k, h, r)
         add ({(k, h, r)}, b, p) to list Ranking sorted
         by b in descending order;
return P_R;
```

Figure 3: Greedy ranking algorithm

The complexity calculations include the complexity of computing the benefit function. They assume the optimal adaptation paths $\pi^{\text{opt}}_{c,k}$ are computed using the Dijkstra algorithm in Γ (cf. Sec. 3.1) accounting for $O(|H|^2|B|^2|R|^2)$.

The above algorithms are based on a static model and do not explicitly take the dynamics in the system into account. They assume perfect knowledge about the load and state of the system. In practice, however, load and system state are neither static nor do we have perfect knowledge about them. Even though, we adjust to dynamic changes by regularly recalculating the placement based on fresh estimates, those estimates are rather inaccurate and become outdated quickly. Nevertheless, we still assume the algorithms to produce reasonable placements because they have been shown to be rather tolerant to imperfect input data when applied to the general RP problem [15]. Qiu et al. [15] prove the cost of a placement produced by a greedy ranking heuristic (for cost-optimized general RP) to be within the factor of 2 of the cost of an optimal placement even if the input data is distorted with a random error of up to 400%. For comparison, with exact input data it is within the factor of 1.1 – 1.5. The experimental proof of the assumption that the heuristics are similarly robust when applied to adaptation-aware RP as well as the evaluation of the goodness of placements produced by the heuristics is subject to ongoing research.

4. RELATED RESEARCH

The finding that the effectiveness of replication techniques may benefit from taking adaptability of media objects into consideration motivated research projects, such as the Soft Caching project [11]. In this project, the researchers experimented with scaling down cached images to free cache memory without eviction. The approach was later extended to quality adaptive caching of layered encoded video [10, 16]. In the context of quality adaptive caching of streaming media, the issue of cost-quality-optimization has also been addressed [17]. Those projects, however, restrict their consideration to a single, autonomous cache. They do not exploit the advantages of replication in a global network of cooperating surrogates in a CDN.

The importance of content adaptation within CDNs has been identified by previous work. Wee et al. [18], for instance, describe the architecture of a Mobile Streaming Media CDN (MSM-CDN) providing for adaptation of media streams within the CDN. Also, the development of the Internet Content Adaptation Protocol (ICAP) [7], which is now under revision and further development by the IETF Open Pluggable Edge Services (OPES) working group, is targeted at allowing for content adaptation in CDNs. Nevertheless, we are not aware of any publications addressing the issue of adaptation-aware RP as dealt with by our research.

Even though adaptation-aware RP in A-CDNs is a new research topic, it is closely related to the field of general RP research. General RP deals with the distribution of replicas where content adaptation is not considered and hence there is only one representation of each content object. Because RP is proven to be NP-hard [12], several heuristic approaches have been evaluated (see [13] for a survey). For capacity constrained RP problems, plain and greedy ranking heuristics have been shown to produce good placements [13, 12]. Greedy algorithms have also been successfully applied in the view selection problem in data warehouses [1, 9], a

problem that is structurally similar to RP in A-CDNs. For a single knapsack problem (one centralized constraint resource) a 63% lower bound on the benfit by a greedy solution compared to the optimal solution has been proven [9]. However, that bound does not hold for the multiple knapsack problem in the distributed case, where every site accounts for a capacity constraint [1].

Besides the RP problem, our paper also relates to previous work on the composition of distributed adaptation paths. The notion of distributed adaptation paths has been introduced by the Ninja project [8] achieving adaptation through automated service chaining. Choi et al. [5] addressed the optimization of adaptation path composition by mapping the problem onto a conventional shortest path search in multi-layered graphs. We adopted this idea in our path composition algorithm. Though Candan et al. [4] do not explicitly refer to shortest path algorithms, their algorithm for object synthesis in collaborative multimedia systems is similar to the ideas of [5]. [4] presents the only path composition algorithm that allows for cost-quality-optimization.

5. CONCLUSION

This paper has dealt with replica placement in Adaptive Content Distribution Networks. In A-CDNs, different representations of an object may be stored in the surrogates and adapted to the clients by the CDN during delivery. Hence, RP in A-CDNs is targeted at deciding which representation of which object to store in which surrogate.

In the first part of the paper, we have introduced a formal model for cost-quality-optimized adaptive content networking and defined the RP optimization problem. The definition of the objective function assumes the optimal adaptation path to satisfy a request to be known. Hence, optimal adaptation path composition is an inherent sub-problem of RP. We have proposed an algorithm to tackle this problem by mapping it onto conventional shortest path search.

The actual RP optimization problem is NP-hard. That is why we favor heuristic approaches. We have presented plain and a greedy ranking algorithm. Ranking approaches have proven successful in similar problems such as general RP or view selection in data warehouses.

The experimental evaluation of the heuristics is subject to ongoing research. We are currently preparing simulations of an A-CDN that distributes Web images and video. Whereas the general distribution of Web requests is well known from previous work, the distribution of specific media features, e.g. spatial and color resolution, had to be determined by an analysis of images available on the Web through a proxy trace. Current work deals with the construction of a reasonable model of different adaptation operations that describes the effect of the operations on the media features and thereby on the quality of adapted content.

6. REFERENCES

[1] A. Bauer and W. Lehner. On solving the view selection problem in distributed data warehouse architectures. In *15th Int'l Conf. on Scientific and Statistical Database Management (SSDBM)*, Cambridge, MA, 2003.

[2] S. Buchholz and T. Buchholz. Adaptive content networking. In *Int'l Symposium on Information and Communication Technology*, Dublin, Ireland, 2003.

[3] S. Buchholz and A. Schill. Adaptation-aware web caching: Caching in the future pervasive web. In *13th GI/ITG Conference Kommunikation in verteilten Systemen (KiVS)*, Leipzig, Germany, 2003.

[4] K. S. Candan, V. S. Subrahmanian, and P. V. Rangan. Collaborative multimedia systems: Synthesis of media objects. *IEEE Transactions on Knowledge and Data Engineering*, 10(3):433–457, 1998.

[5] S. Choi, J. Turner, and T. Wolf. Configuring sessions in programmable networks. In *IEEE INFOCOM 2001*, Anchorage, AL, 2001.

[6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2000.

[7] J. Elson and A. Cerpa. Internet Content Adaptation Protocol (ICAP), RFC 3507, April 2003.

[8] S. D. Gribble, M. Welsh, R. von Behren, E. A. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. D. Joseph, R. H. Katz, Z. M. Mao, S. Ross, and B. Zhao. The ninja architecture for robust internet-scale systems and services. *Journal of Computer Networks*, 35(4), 2001.

[9] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, 1996.

[10] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross. Distributing layered encoded video through caches. In *IEEE INFOCOM 2001*, Anchorage, AL, 2001.

[11] J. Kangasharju, Y. Kwon, and A. Ortega. Design and implementation of a soft caching proxy. In *3rd Int'l WWW Caching Workshop (WCW)*, Manchester, UK, 1998.

[12] J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in content distribution networks. In *6th Int'l Workshop on Web Content Caching and Distribution (WCW)*, Boston, MA, 2001.

[13] M. Karlsson, C. Karamanolis, and M. Mahalingam. A framework for evaluating replica placement algorithms. Technical Report HPL-2002-219, HP Labs, 2003.

[14] G. Klyne. A syntax for describing media feature sets. RFC 2533, 1999.

[15] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *IEEE INFOCOM 2001*, Anchorage, AL, 2001.

[16] R. Rejaie, H. Yu, M. Handley, and D. Estrin. Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet. In *IEEE INFOCOM 2000*, Tel Aviv, Israel, 2000.

[17] A. Sehgal and P. A. Chou. Cost-distortion optimized caching of streaming media. In *IEEE Int'l Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Orlando, FL, 2002.

[18] S. Wee, J. Apostolopoulos, W. Tan, and S. Roy. Research and design of a mobile streaming media content delivery network. In *IEEE International Multimedia Conference and Expo (ICME)*, Baltimore, MD, 2003.