Evaluation of Replica Placement and Retrieval Algorithms in Self-Organizing CDNs

Jan Coppens, Tim Wauters, Filip De Turck, Bart Dhoedt and Piet Demeester
Department of Information Technology (INTEC), Ghent University - IMEC
Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
Tel.: +32 9 33 14974, Fax: +32 9 33 14899
E-mail: Jan.Coppens@intec.ugent.be

***Abstract*— Content distribution networks (CDN) have been increasingly used as the scalable solution to deliver high-quality multimedia content. However, despite the promising concept, current self-organizing content distribution networks (using real-time replica placement) lack placement and retrieval algorithms that are both intelligent and scalable. In this paper we present an architecture and relating algorithms for such a scalable, self-optimizing content delivery service. Contrary to distributed and centralized CDN architectures, we propose a hybrid approach. In order to tackle specific problems such as congested network parts and the occurrence of flash crowds, network monitoring and multi-source traffic engineering are combined.**

I. Introduction

In the classical client-server architecture, a single video server serves multiple clients. This approach however has some significant drawbacks. The quality of the offered services often degrades when for example the server cannot handle the load, the intermediate network gets congested and starts dropping packets or the experienced latency is too high. To tackle the performance problems of this classical client-server approach, Content Distribution Networks [1] have been introduced. In these CDNs, the content is replicated to different surrogate servers at the edges of the network. This way, the content only has to pass a few nodes in order to reach the end user, resulting in better Quality of Service (QoS) and network usage. The CDN will use a Replica Placement Algorithm (RPA) [2] to decide which content to replicate on which server. Likewise, Content Retrieval Algorithms are used to direct client requests to an optimal server.

However, the main problem in the current CDNs is that both algorithms rely in most cases on some raw network measurements such as round-trip-times and hop-counts. These measurements do not provide information on the available and used bandwidth, one-way delay and lossrate of network paths. Existing CDNs only have a limited view on the current state of the transport network resources and content servers, resulting in non-optimal decisions of the current algorithms ([3][4]). As shown in [2] centralized RPAs offer the best placement, while only distributed algorithms can scale to larger networks. In this paper we propose a hybrid architecture and complementing RPA algorithms, called COCOA, which are based on dynamic and as precise as possible information on the current state of the network. It will be shown that COCOA achieves a similar performance as current centralized algorithms, while maintaining the scalability of distributed heuristics.

The paper is structured as follows. Other work related to this research is described in section II. Section III briefly illustrates the developed CDN architecture with respect to the integrated self-organizing algorithms. Section IV describes the COCOA RPA, while section V compares its performance to other algorithms. Finally, section VI presents a conclusion and future work.

II. Related Work

Distribution and replication of data is used frequently to improve the availability and performance of various types of services. L. Dowdy and D. Foster introduced the concept in the file assignment problem [5]. In [6], the authors use centralized placement strategies for Web server replicas, while [7] describes some basic replication techniques in general content distribution networks. More advanced centralized and distributed replica placement algorithms are defined and compared in [2] and [8]. All previous approaches try to optimize the overall system performance by minimizing the access latency and/or the required replication resources (i.e. server space). A slightly different approach in [9] guarantees certain latency goals while minimizing the replication cost.

Contrary to the scalable distributed and more optimal centralized placement algorithms, we propose a hybrid system architecture in which centralized components are combined with distributed placement algorithms. While previous papers consider optimization metrics such as access latency, we minimize the load of the core network in function of the replication cost. This allows network operators to dimension their network according to the user demand. To validate our novel content placement and retrieval algorithms, we compare them with the algorithms described in [2] and [7].

III. Overview of the CDN Architecture

In order to facilitate the distribution of content (constant bitrate streaming audio/video), a novel architecture is designed. This architecture consists of different functional layers, each containing autonomous modules in order to provide portability and extensibility of the framework. Layers and modules communicate through a predefined interface. Because

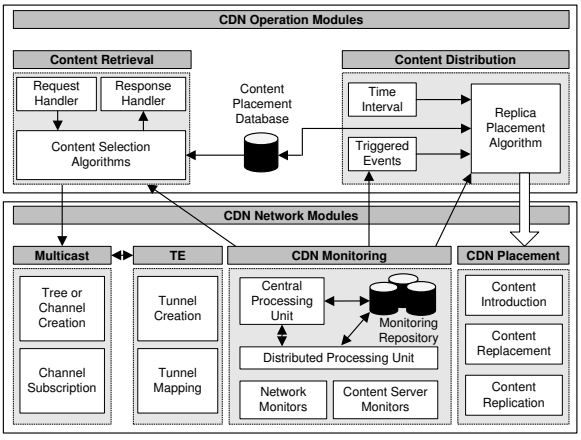


Fig. 1. Snapshot of the CDN Architecture

of the modular design, it is possible to include new platforms, hardware, placement algorithms etc.

Figure 1 focusses on the part of the architecture that is relevant for this discussion. The CDN Operation layer hosts the logic of the CDN platform. Content retrieval and replica placement algorithms in the modules of this layer use the functionality of the CDN Network Management layer to implement their decisions in the network. This layer contains modules that communicate with the individual network components such as servers, routers and switches in the bottom CDN Hardware layer. Because RPAs are invoked real-time once every period, the CDN will self-organize over time.

To meet the advantages of the scalable distributed and more optimal centralized content distribution networks, this CDN architecture uses a hybrid approach. In this hybrid CDN, the Content Retrieval (CR) module is centralized, while the Content Distribution (CD) module is distributed. Because centralized replica placement algorithms can take more information on the state of the network into account (e.g. global topological view), their produced placement is closer to the exact solution than the one produced by distributed heuristics. Based on this observation, the centralized CR module in the presented architecture provides additional network state information to the distributed CD modules, resulting in a more accurate content placement. For each content request, the CDN needs to find a server and a path to the client, independently whether the RPA runs centralized or distributed[1]. The CR module reuses the computed path cost and passes it to the participating content servers. This way, the overhead of the centralized module is limited to the distribution of the obtained information.

Both the CR and CD modules obtain network state information from the CDN monitoring module. This module feeds the different algorithms in the upper layers with measured network state such as topology information and available bandwidth, delay, loss and jitter on the core edges. In order to obtain this state information, multiple observation points are configured in the network. The monitoring platform and implemented CDN monitoring module are fully addressed in [10].

[1]This can be compared to dynamic DNS request routing [1].

IV. Replica Placement and Content Retrieval Algorithms

The most vital parts of the architecture are the algorithms that place the content and direct clients to the most optimal replica. These algorithms will determine the performance of the CDN architecture. In the next sections different placement algorithms are covered and compared to the exact solution. A novel algorithm that combines the advantages of existing RPAs will be described and evaluated.

A. Reference solution by means of an ILP formulation

In order to determine the optimal placement of the content replicas in a given situation (known requests), an Integer Linear Program (ILP) can be evaluated off-line. Because our replica placement ILP is proven to be NP-complete, it is not suited for real-time CDN environments. The following formulation will be used as a reference to evaluate the RPA heuristics.

For a set of edges E, users U and files F, we define the bitrate b_f of file f and the number of requests $r_{u,f}$ from user u for file f. I_n is the set of all incoming edges, while O_n defines the set of all outgoing edges of node n. A node n belongs to the set of users U, servers S or core nodes C. We minimize the average link load in order to serve as much requests as possible:

$$\texttt{min}\left(\frac{\sum_{e\in E}\left(\frac{\sum_{u\in U}\left(\sum_{f\in F}\left(h_{e,u,f}\times b_f\times r_{u,f}\right)\right)}{\texttt{capacity}_e}\right)}{\#E}\right) \qquad (1)$$

$h_{e,u,f}$ is 1 if edge e is used to deliver file f to user u, 0 otherwise.
$z_{s,f}$ is 1 if file f is stored on server s, 0 otherwise.

The ILP is subject to the flow conservation constraints:

$$\begin{aligned}
&\textstyle\sum_{e\in I_c} h_{e,u,f} = \sum_{e\in O_c} h_{e,u,f} && \forall u\in U, f\in F, c\in C\\
&z_{s,f} \geq h_{e,u,f} && \forall s\in S, u\in U, f\in F, e\in O\\
&\textstyle\sum_{e\in I_u} h_{e,u,f} = r_{u,f} && \forall u\in U, f\in F
\end{aligned}$$

The previous constraints ensure that a flow starts at a server, flows through the network and reaches a client. Depending on the scenario, other constraints that limit the edge, server or access capacity, limit the number of replicas and enable/disable traffic engineering are applied as well.

B. General real-time replica placement heuristics

The most simple on-line RPA algorithm is *randomly* distributing content replicas. Despite the fact that this algorithm is very easy to implement, it will not make an optimal decision on the number and location of the replicas. The *popularity-local* (pop-L) algorithm will store the content that is locally most popular. Each content server stores the most popular content among its clients. The *popularity-global* (pop-G) RPA on the other hand extends the popularity-local algorithm by putting a (probabilistic) limit on the number of replicas of the

most popular content in favor of other content. This way, not all servers are filled with the same most popular content.

The pop-L algorithm makes a decision based on the number of received requests independently of the distance to other replicas. Greedy algorithms do take this distance into account. The *greedy-single* RPA (gre-S) for example places its content based on the popularity and the cost of retrieving content from the origin server, while the *greedy-global* (gre-G) algorithm depends on the popularity and the cost of retrieving content from other servers [7]. Optimizing the placement even further, *greedy-all* (gre-A) computes the optimal position of the content based on the local popularity and the cost of retrieving content from other servers by all clients.

C. Co-Operative Cost Optimization Algorithm (COCOA)

COCOA is a heuristic for replica placement designed to fit the presented CDN architecture. It aims to combine the benefits of both the popularity en greedy algorithms. COCOA relies on the co-operation between both the CR and CD modules, i.e. the CR module will aid the decision-making process of the CD module. Furthermore, servers will co-operate when retrieving content. Unlike the greedy alternatives, COCOA is a progressive RPA, which means that not all replica placements are recomputed at each RPA iteration. When the RPA is executed, only the best placements are computed and stored in favor of already present content. This calls for the need of a content removal strategy.

a) The CR Module: For each request, the CR algorithm computes the cost (e.g. hopcount) of the best or second best (in case a replica is stored on the local server) path from a candidate streaming server to the client[2].

> Define: Server $s \in S$ with content C_s; Requests $r_{s,f}$ for content f from server s; $Cost_f^{s',s}$ in order to stream content f from server s' to s; Local Cost $LC_{s,f}$ of content f in server s; Local Profit $LP_{s,f}$ of content f in server s
>
> Find s' so that $Cost_f^{s',s} \leq Cost_f^{s'',s}$, with $f \in (C_{s'} \cap C_{s''}), \forall s' \neq s, \forall s'' \neq s, \forall r_{s,f}$
> If no such s' can be found, then $Cost_f^{s',s} = Cost_{MAX}$
> If $(f \notin C_s)$, then $LC_{s,f} = LC_{s,f} + Cost_f^{s',s}$
> else $LP_{s,f} = LP_{s,f} + Cost_f^{s',s}$

If the content is not present locally, we increase the cost with the cost we want to avoid by storing a replica on the local server. If the content on the other hand is present locally, we increase the profit with the hypothetical cost we would experience when the content is removed locally. The smaller this cost, the fewer the profit of this copy increases and the faster the content will be removed in favor of other content (i.e. the content is replicated in the vicinity of this server, so it can be removed without any severe performance hits). If the content is not stored on other servers, i.e. the content can not be removed, the profit is increased with the maximum cost possible (e.g. max hopcount). Depending on the different simulations, the cost will be expressed as distance (hopcount) or path/network load. For each request the CR algorithm needs to find at maximum one server-path pair, i.e. the optimal path in case the content is not present locally, the semi-optimal path in case the content is present locally and somewhere else or none.

b) The CD Module: Executing the COCOA RPA on a single server s will search for expensive content f_{in} (i.e. content that is not present and has the highest local cost $LC_{s,f_{in}}$). If the server has spare storage capacity, this content is stored. In the case the server is already fully loaded, we search for other content f_{out} that can be removed (i.e. content that is locally present and has the lowest profit $LP_{s,f_{out}}$). If the profit of this latter content f_{out} is lower than the cost of the prior content f_{in}, f_{out} is removed in favor of f_{in} (we assume all content has the same size). When storing or removing content, the values for local profit LP and cost LC are swapped. When evaluating the COCOA RPA, only the cost functions for different files need to be compared. This results is a high performance and allows more content and bigger networks.

[2]Note that finding a server and path is necessary for every request, independently of the used RPA.

TABLE I
CHARACTERISTICS OF VARIOUS RPA HEURISTICS

RPA	Requests	Topo	Process	Complexity
Random	None	None	Distr	$O(C_sS)$
Pop-L	Local	None	Distr	$O(C_sSF)$
Pop-G	Global	None	Hybrid	$O(C_sSF)$
Gre-S	Local	Origin	Distr	$O(C_sSF)$
Gre-G	All	Entire	Centr	$O(C_{all}S^2F)$
Gre-A	All	Entire	Centr	$O(C_{all}S^3F)$
COCOA	CR Mod	None	Hybrid	$O(C_sSF)$

V. EVALUATION OF THE RPA HEURISTICS

A. Complexity and scalability

Table I illustrates the characteristics of the various considered RPA heuristics. When executing an RPA only local or all requests are taken into account. In addition to the local requests, the pop-G algorithm needs to know the global number of all generated requests. Because COCOA is part of the developed CDN architecture, it depends on the CR module to present the required request information. Popularity algorithms do not need any topological information, while the 2 greediest RPAs need to know the cost of every network path. The gre-S RPA only considers the cost from the origin server. Both previous characteristics define whether an RPA runs distributed or centralized. A hybrid RPA on the other hand combines both approaches. The complexity of an algorithm reflects the time of a single execution. S defines the number of servers, while F is the total amount of different content. The average number of files that can be stored on a single server is defined by C_s, the total amount of storage space by C_{all}. For distributed and hybrid algorithms, S can be omitted from the processing time because of parallel execution.

Ignoring the parallelism which is inherent to distributed algorithms, the processing times of the various RPAs even differs in orders of magnitude. Using our developed Java finite-state simulator ($S = 28$, $F = 2800$, $C_s = 200$, $C_{all} = C_s \times S$), running the pop-L or COCOA RPA lasts about 0.2 seconds, where the gre-G RPA takes an average of 38 seconds. Even worse, to calculate all new replica positions using the gre-A heuristic, an average time of 15.9 minutes is required.

The presented CDN architecture would benefit most from the greedy algorithms because they result in the best content placement. However, these algorithms are not suited to run in a large CDN. Computing greedy algorithms is very time and processor consuming and scales very bad with the size (i.e. number of servers, content and requests) of a CDN. Each time a greedy algorithm runs, it recomputes the position of every replica, which is impossible in a CDN with a large number of files. Ideally, we need an RPA that gives a solution as optimal as a greedy algorithm, but with the complexity of the popularity algorithm.

B. Performance analysis of the replica placement algorithms

To evaluate the performance of the various placement algorithms, they were implemented in a Java finite-state simulator.

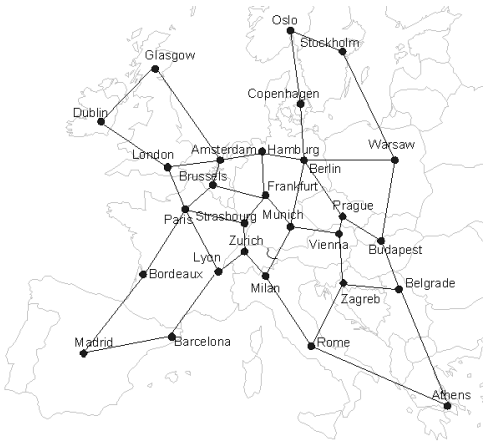


Fig. 2. Pan-European fiber-optic network topology

Figure 2 shows the used topology of a realistic pan-European fiber-optic network [11], where each edge has a throughput of 1Gb. The popularity of the content is distributed zipf-like (α = 0.7) [12] and $F = 105$ (streamed at a rate of 2Mbit/s for 120'). In this scenario approximately 2 movies are requested each minute ($R = 2$). The parameters are considered reasonably low in order to be able to use them in the ILP solution.

Figure 3 depicts the increased load of the various RPA heuristics relative to the exact solution in function of the replication factor RF, i.e. total additional storage capacity as a percentage of the available content. In all scenarios the RPA is executed every 50 minutes. Clearly, the placement provided by the COCOA algorithm is much better than the pop-L heuristic. The solution from COCOA comes very close

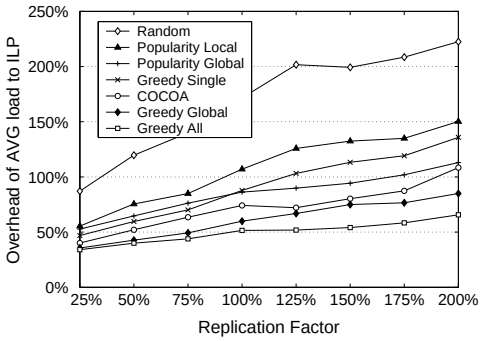


Fig. 3. Comparison of the average load of the heuristics with the ILP

to the one offered by gre-G, but only for a fraction of the cost. The gre-A RPA reduces the load even further, however its computational complexity makes it unscalable to larger networks.

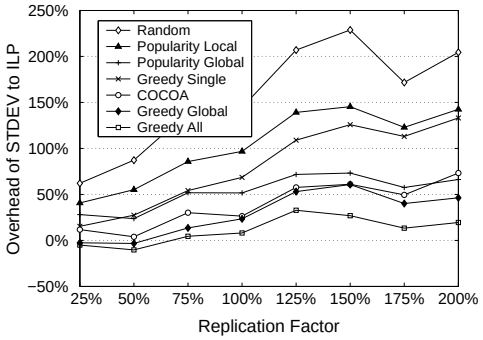


Fig. 4. Comparison of the standard deviation of the heuristics with the ILP

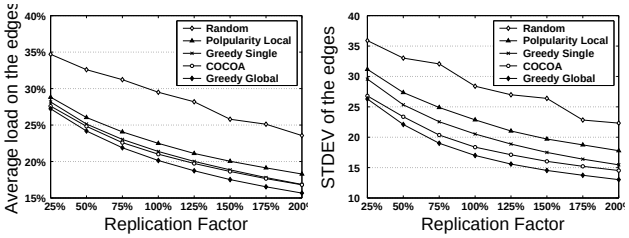
In Figure 4 the spread of the load in the network is considered for the various algorithms. A low standard deviation means that the load is equally spreaded over the network, resulting in a more efficient use of the available bandwidth. Again COCOA performs almost as good as the gre-G RPA.

Each time the RPA is executed (every 50 minutes), content will be removed in favor of other content, resulting in network overhead. For both the popularity and COCOA algorithms, the number of replacements is similar. An RF of 200% for example, results in 200 to 300 replacements when the RPA is executed 200 times over a period of 7 days. For the same number of executions the greedy algorithms will replace about 950 files, resulting in a much higher network overhead.

C. Using traffic engineering for load balancing

A major issue in CDN networks is the occurrence of flash crowds and congestion. In a flash crowd, the current request rate raises far above the predicted or historical rate. Flash crowds drastically change the popularity of certain content globally as well as locally. This has severe repercussions on the load and can cause congestion in certain parts of the network. Because flash crowds are difficult to anticipate, preemptive measures need to be taken. In order to be as resilient as

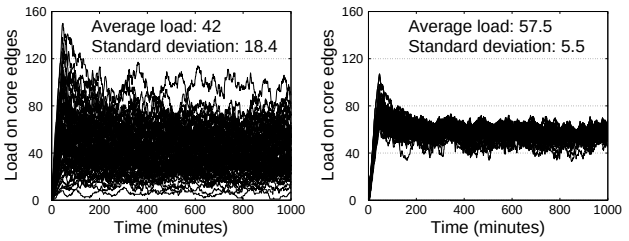
possible the RPA and CR algorithm can be tuned to spread the load over the entire network. Instead of directing the client to the closest replica, the client is directed to the replica that has a path to the client resulting in the lowest overhead on all edges of that path, i.e. pick a server and path so that the maximum of the load on all edges of the path is lower than the maximum load of all other server-path combinations [13]. Using this multi-source traffic engineering (TE), the CR module avoids all links that are heavily loaded.



(a) AVG load (b) STDEV

Fig. 5. Average load and STDEV without TE

Without the use of TE, all flows are mapped to the shortest path from server to client. For a CDN with the same parameters as before, except $F = 2800$ and $R = 15$, figures 5 (a) and (b) plot the average load and standard deviation of all edges for various RPAs. Again COCOA scores better than pop-L and slightly worse than gre-G. Figure 5 (b) shows that the more intelligent algorithms not only reduce the load, but also spread it more equally over all edges.

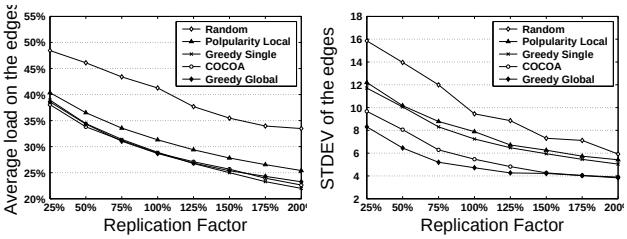


(a) Without TE (b) With TE

Fig. 6. Effect of TE on the load (COCOA with $RF = 200\%$)

When flows are mapped to paths other than the shortest path, the average path length will increase, but all edges will be loaded more equally. Figure 6 clearly shows the effect of this preemptive TE.

Figure 7 illustrates the results of the same experiments as figure 5 with the use of multi-source TE. Even though the average link load increases, the load on the most loaded core edges drops up to 30%. By diverting traffic from these links to other network paths, congestion is reduced. With preemptive TE, again COCOA scores better than pop-L. Because the COCOA algorithm takes the edge load into consideration (by means of the CR module), it can optimize the content placement even further. This results in a placement that is as optimal as the one obtained by gre-G.



(a) AVG load (b) STDEV

Fig. 7. Average load and STDEV with TE

VI. Conclusion and Future Work

This paper briefly presented a novel hybrid CDN architecture and related COCOA placement and retrieval algorithms. Because centralized components are mixed with distributed algorithms, COCOA is as scalable as the popularity-local algorithm, while providing a similar performance as the greedy-global RPA. Up until now it is assumed that the number of content replacements has no influence on the load of the network. In a realistic situation, these replacements need to be kept to a minimum. In future work, we will investigate the effect of the frequency of consecutive RPA executions and the number of content replacements and search for algorithms in order to optimize this even further.

References

[1] D. C. Verma, *Content Distribution Networks: An Engineering Approach*. New York: John Wiley & Sons, Inc., 2002.

[2] M. Karlsson and M. Mahalingam, "Do we need replica placement algorithms in content delivery networks?" in *Seventh International Web Content Caching and Distribution Workshop*, Aug 2002.

[3] K. Johnson, J. Carr, M. Day, and M. Kaashoek, "The measured performance of content distribution networks," in *5th International Web Caching and Content Delivery Workshop*, 2000.

[4] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proceedings of IEEE INFOCOM'02*, 6 2002.

[5] L. Dowdy and D. Foster, "Comparative models of the file assignment problem," in *ACM Computer Surveys (CSUR)*, June 1982.

[6] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *Proceedings of IEEE Infocom*, Apr 2001.

[7] J. Kangasharju, J. Roberts, and K. Ross, "Object replication strategies in content distribution networks," in *Proceedings of Sixth International Workshop on Web Caching and Content Delivery*, Jun 2001.

[8] M. Karlsson, C. Karamanolis, and M. Mahalingam, "A framework for evaluating replica placement algorithms," in *Technical Report, HP Laboratories*, Jul 2002.

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

[10] J. Coppens, E. Markatos, J. Novotny, M. Polychronakis, V. Smotlacha, and S. Ubik, "Scampi - a scalable monitoring platform for the internet," in *International Workshop on Inter-Domain Performance and Simulation (IPS)*, Budapest, Hungary, Mar 2004.

[11] S. D. Maesschalck, "Pan-european optical transport networks: an availability-based comparison," *Photonic Network Communications*, vol. 5, no. 3, pp. 203–225, May 2003.

[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE Infocom*, 1999.

[13] T. Wauters, J. Coppens, B. Dhoedt, and P. Demeester, "Load balancing through efficient distributed content placement," in *1st EuroNGI Conference on Next Generation Internet Networks - Traffic Engineering*, Rome, Apr 2005.