This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

CDNs Content Outsourcing via Generalized Communities

Dimitrios Katsaros[1,2], George Pallis[1,3], Konstantinos Stamos[1], Athena Vakali[1], *Member, IEEE*
Antonis Sidiropoulos[1], Yannis Manolopoulos[1], *Member, IEEE*

***Abstract*—Content Distribution Networks (CDNs) balance costs and quality in services related to content delivery. Devising an efficient content outsourcing policy is crucial since, based on such policies, CDN providers can provide client-tailored content, improve performance, and result in significant economical gains. Earlier content outsourcing approaches may often prove ineffective since they drive prefetching decisions by assuming knowledge of content popularity statistics, which are not always available and are extremely volatile. This work addresses this issue, by proposing a novel self-adaptive technique under a CDN framework on which outsourced content is identified with no a-priori knowledge of (earlier) request statistics. This is employed by using a structure-based approach identifying coherent clusters of "correlated" Web server content objects, the so-called Web page communities. These communities are the core outsourcing unit and in this paper a detailed simulation experimentation has shown that the proposed technique is robust and effective in reducing user-perceived latency as compared with competing approaches, i.e., two communities-based approaches, Web caching and non-CDN.**

***Index Terms*—Caching, replication, Web communities, content distribution networks, social network analysis.**

I. Introduction

Distributing information to Web users in an efficient and cost-effective manner is a challenging problem, especially, under the increasing requirements emerging from a variety of modern applications, e.g., voice-over-IP, streaming media, etc. Eager audiences embracing the "digital lifestyle" are requesting greater and greater volumes of content on a daily basis. For instance, the Internet video site YouTube hits more than 100 million videos per day[1]. Estimations of YouTube's bandwidth go from 25TB/day to 200TB/day. At the same time, more and more applications (such as e-commerce, e-learning etc.) are relying on the Web but with high sensitivity to delays. A delay even of a few milliseconds in a Web server content may be intolerable. At first, solutions such as Web caching and replication were considered as the key to satisfy such growing demands and expectations. However, such solutions (e.g., Web caching) have become obsolete due to their inability to keep up with the growing demands and the unexpected Web-related phenomena such as the flash-crowd events [17] occurring when numerous users access a Web server content simultaneously (now often occurring on the Web due to its globalization and wide adoption).

[1]Department of Informatics, Aristotle University, Thessaloniki, Greece.
[2]Department of Computer & Communication Engineering, University of Thessaly, Volos, Greece.
[3]Department of Computer Science, University of Cyprus, Nicosia, Cyprus.
[1]http://www.youtube.com/

CDNs have been proposed to meet such challenges by providing a scalable and cost-effective mechanism for accelerating the delivery of the Web content [7], [27]. A CDN[2] is an overlay network across Internet (Figure 1), which consists of a set of surrogate servers (distributed around the world), routers and network elements. Surrogate servers are the key elements in a CDN, acting as proxy caches that serve directly cached content to clients. They store copies of identical content, such that clients' requests are satisfied by the most appropriate site. Once a client requests for content on an origin server (managed by a CDN), his request is directed to the appropriate CDN's surrogate server. This has a result to improve both the response time (the requested content is nearest to the client) and the system throughput (the workload is distributed to several servers).

As emphasized in [4], [34], CDNs significantly reduce the bandwidth requirements for Web service providers, since the requested content is closer to user and there is no need to traverse all of the congested pipes and peering points. So, reducing bandwidth reduces cost for the Web service providers. CDNs provide also scalable Web application hosting techniques (such as edge computing [10]) in order to accelerate the dynamic generation of Web pages; instead of replicating the dynamic pages generated by a Web server, they replicate the means of generating pages over multiple surrogate servers [34].

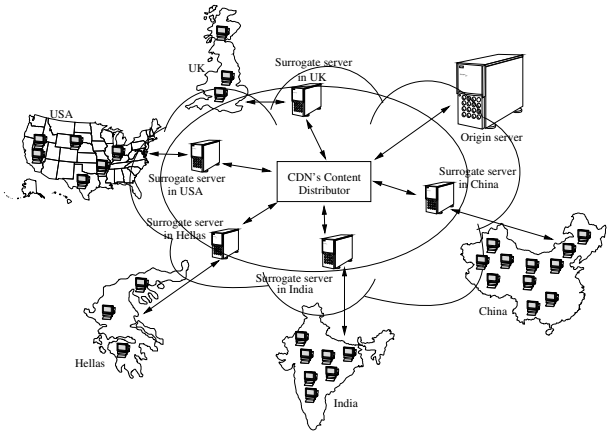


Fig. 1. A typical Content Distribution Network.

CDNs are expected to play a key role in the future of the Internet infrastructure since their high end-user performance

[2]A survey on the status and trends of CDNs is given in [36]. Detailed information about the CDNs' mechanisms are presented in [30].

Digital Object Indentifier 10.1109/TKDE.2008.92

and cost savings have urged many Web entrepreneurs to make contracts with CDNs[3]. This trend is also reflected by the stock market since for instance, Akamai[4] (a key leader in the CDN providers) stock price has gone up 260% in the past 12 months, and its market capitalization currently tops $8.8 billion[5].

A. Motivation and paper's contributions

Currently CDNs invest in large-scale infrastructure (surrogate servers, network resources etc.), to provide high data quality for their clients. To revenue their investment, CDNs charge their customers (i.e., Web server owners) based on two criteria : the amount of content which has been outsourced and their traffic records (measured by the content delivery from surrogate servers to clients). According to a quite recent CDN market report[3], the average cost per GB of streaming video transferred in 2004 through a CDN was $1.75, while the average price to deliver a GB of Internet radio was $1. Given that the bandwidth usage of Web servers content may be huge, (i.e., the bandwidth usage of YouTube is about 6 Petabytes per month), it is evident that this cost may be extremely high.

Therefore, the proposal of a content outsourcing policy, which will reduce both the Internet traffic and the replicas maintenance costs, is a challenging research task due to *the huge-scale, the heterogeneity, the multi levels in the structure, the hyperlinks and interconnections, the dynamic and evolving nature* of Web content.

To the best of the authors' knowledge, earlier approaches for content outsourcing on CDNs assume knowledge of content popularity statistics to drive the prefetching decisions [9], giving an indication of the popularity of Web resources (a detailed review of relevant work will be presented in Section II). Such information though, is not always available, or it is extremely volatile, turning such methods problematic. The use of popularity statistics has several drawbacks. Firstly, it requires quite a long time to collect reliable request statistics for each object. Such a long interval though may not be available, when a new site is published to the Internet and should be protected from "flash crowds" [17]. Moreover, the popularity of each object varies considerably [4], [9]. In addition, the use of administratively tuned parameters to select the hot objects causes additional headaches, since there is no apriori knowledge about how to set these parameters. Realizing the limitations of such solutions, authors in [30] implied the need for self-tuning content outsourcing policies. In [32], we initiated the study of this problem by outsourcing clusters of Web pages. The outsourced clusters are identified by naively exploring the structure of the Web site. Results showed that such an approach improves the CDN's performance in terms of user-perceived latency and data redundancy.

The present work continues and improves upon the authors' preliminary efforts in [32] focusing on devising a high performance outsourcing policy under a CDN framework. In this context we point out that the following challenges are involved:

- outsource objects that should be popular for long time periods;
- refrain from using (locally estimated or server-supplied) tuneable parameters (e.g., number of clusters) and keywords, which don't adapt well to changing access distributions;
- refrain from using popularity statistics which don't represent effectively the dynamic users' navigation behaviour. As observed in [9], only 40% of the "popular" objects of the one day remain "popular" and the next day.

In accordance to the above challenges, we propose a novel self-adaptive technique under a CDN framework on which outsourced content is identified by exploring Web server content structure and with no a-priori knowledge of (earlier) request statistics. This paper's contribution is summarized in:

- *identifying content clusters*, called Web page communities, based on the adopted Web graph structure (where Web pages are nodes and hyperlinks are edges), such that these communities serve as the core outsourcing unit for replication. Typically, it can be considered as a dense sub-graph where the number of edges within community is larger than the number of edges between communities. Such structures exist on the Web [1], [13], [20] — Web servers content designers (humans or applications) tend to organize sites into collections of Web pages related to a common interest — and affect users' navigation behavior; a dense linkage implies a higher probability of selecting a link. Here, we exploit a quantitative definition for Web page communities introduced in [32], which is considered to be suitable for CDNs content outsourcing problem. Our definition is flexible, allowing overlaps among communities (a Web page may belong to more than one community), since a Web page usually covers a wide range of topics (e.g. a news Web server content) and cannot be classified by a single community. The resulted communities are entirely being replicated by the CDN's surrogate servers.
- *defining a parameter-free outsourcing policy*, since our structure-based approach (unlike k-median, dense k-subgraphs, min-sum or min-max clustering), does not require the number of communities as a predetermined parameter, but instead, the optimal number of communities is any value between 1 and the number of nodes of the Web site graph, depending on the node connectivity (captured by the Web server content structure). The proposed policy, called CiBC (Communities identification with Betweenness Centrality), identifies overlapped Web page communities using the concept of Betweenness Centrality (BC) [5]. Specifically, authors in [22] have used the concept of edge betweenness to select edges to be removed from the graph so as to devise a hierarchical agglomerative clustering procedure, which though is not capable of providing the final communities but requires intervening of administrators. Contrary to this work [22], the BC is used, in this paper, to measure how central each node of the Web site graph is within a community.
- *experimenting on a detailed simulation testbed*, since the

[3]"Content Delivery Networks, Market Strategies and Forecasts (2001-2006)", AccuStream iMedia Research:http://www.researchandmarkets.com/

[4]Akamai http://www.akamai.com/

[5]http://uk.finance.yahoo.com/q?s=AKAM

experimentation carried out involves numerous experiments to evaluate the proposed scheme under regular traffic and under flash crowd events. Current usage of Web technologies and Web server content performance characteristics during a flash crowd event are highlighted and from our experimentation the proposed approach is shown to be robust and effective in minimizing both the average response time of users' requests and the costs of CDNs' providers.

B. Road map

The rest of this paper is structured as follows. Section II, discusses the related work. In Section III, we formally define the problem addressed in the paper. Section IV presents the proposed policy. Sections V and VI present the simulation testbed, examined policies and performance measures. Section VII evaluates the proposed approach, and finally, Section VIII concludes the paper.

II. Relevant work

A. Content outsourcing policies

As identified by earlier research efforts [9], [15], the choice of the outsourced content has a crucial impact in terms of CDN's pricing [15] and CDN's performance [9], and it is quite complex and challenging, if we consider the dynamic nature of the Web. A naive solution to this problem is to outsource all the objects of the Web server content (full-mirroring) to all the surrogate servers. The latter may seem feasible, since the technological advances in storage media and networking support have greatly improved. However, the respective demand from the market greatly surpasses these advantages. For instance, after the recent agreement between Limelight Networks[6] and YouTube, under which the first company is adopted as the content delivery platform by YouTube, we can deduce, since this is proprietary information, the huge storage requirements of the surrogate servers. Moreover, the evolution towards completely personalized TV (e.g., the stage6[7]) reveals that the full content of the origin servers can not be completely outsourced as a whole. Finally, the problem of updating such a huge collection of Web objects is unmanageable. Thus, we have to resort to a more "selective" outsourcing policy.

A few such content outsourcing policies have been proposed in order to identify which objects to outsource for replicating to CDNs' surrogate servers. These can be categorized as follows:

- **Empirical-based outsourcing:** The Web server content administrators decide empirically about which content will be outsourced [3].
- **Popularity-based outsourcing:** The most popular objects are replicated to surrogate servers [37].
- **Object-based outsourcing:** The content is replicated to surrogate servers in units of objects. Each object is replicated to the surrogate server (under the storage constraints) which gives the most performance gain (greedy approach) [9], [37].
- **Cluster-based outsourcing:** The content is replicated to surrogate servers in units of clusters [9], [14]. A cluster is defined as a group of Web pages which have some common characteristics with respect to their content, the time of references, the number of references, etc.

From the above content outsourcing policies, the object-based one achieves high performance [9], [37]. However, as pointed out by the authors of these policies, the huge amount of objects results in not being implemented on a real application. On the other hand, the popularity-based outsourcing policies do not select the most suitable objects for outsourcing, since the most popular objects remain popular for a short time period [9]. Moreover, they require quite a long time to collect reliable request statistics for each object. Such a long interval though may not be available, when a new Web server content is published to the Internet and should be protected from flash crowd events.

Thus, we resort to exploit action of cluster-based outsourcing policies. The cluster-based one has also gained the most attraction in the research community [9]. In such an approach, the clusters may be identified by using conventional data clustering algorithms. However, due to the lack of a uniform schema for Web documents and dynamics of Web data, the efficiency of these approaches is unsatisfactory. Furthermore, most of them require administratively tuned parameters (maximum cluster diameter, maximum number of clusters) to decide the number of clusters, which causes additional problems, since there is no a priori knowledge about how many clusters of objects exist and of what shape these clusters are.

In disaccordance with the above approaches, we exploit the Web server content structure and consider each cluster as a Web page community, where its characteristics are that it reflects the dynamic and heterogeneity nature of the Web. Specifically, it considers each page as a whole object, rather than breaking down the Web page into information pieces and reveals mutual relationships among the concerned Web data.

B. Identifying Web page communities

In the literature there are several proposals for identifying Web page communities [13], [16]. One of the key distinguishing properties of the algorithms that is usually considered has to do with the degree of locality which is used for assessing whether or not a page should be assigned in a community. Regarding this feature, the methods for identifying the communities can be summarized as follows:

- **Local-based methods:** These methods (also known as bibliographic methods) attempt to identify communities by searching for similarity between pairs of nodes. Thus, their object is to answer the question "Are these two nodes similar?" In this context, the bibliographic approaches use two similarity metrics: the co-citation and the bibliographic coupling. These techniques, although well-established, fail to find large-scale Web page communities of the Web site graph because the localized structures are too small. More details on their application can be found in [8].

[6]http://www.limelightnetworks.com
[7]http://stage6.divx.com

- **Global-based methods:** These methods (also known as spectral methods) consider all the links in the Web graph (or sub-graph). Specifically, spectral methods identify the communities by removing an approximately minimal set of links from the graph. In this context the Kleinberg's HITS [19] algorithm (stands for Hyperlink-Induced Topic Search) and the PageRank [6] are used to identifying some key nodes of the graph which are related to some community and work well as seed sets [1]. However, without auxiliary text information, both PageRank and HITS have limited success in identifying Web page communities [13]. A well-known global-based approach is the maximum flow one [12]. Given two vertices of a graph G, s and t, the maximum flow problem is to find the maximum flow that can be routed from s to t while obeying all capacity constraints with respect to G. A feature of the flow-based algorithms is that they allow a node to be in at most one community. However, this is a severe limitation in CDNs content outsourcing, since some nodes may be left outside of every community. Another algorithm is the Clique Percolation Method (CPM) [24] which is based on the concept of k-clique community. Specifically, a k-clique community is a complete sub-graph of size k, allowing overlaps between the communities. CPM has been widely used in bioinformatics [24] and in social network analysis [25] and is considered as the state-of-the-art overlapping community finding method.

III. Outsourced content selection

In this paper, we study the problem: *Which content should be outsourced by the origin server to CDN's surrogate servers so as to reduce the load on the origin server and the traffic on the Internet, and ultimately improve response time to users as well as reduce the data redundancy to the surrogate servers?* Next we formally define the problem; the paper's notation is summarized in Table I.

A. Primary problem statement

CDNs are overlay networks on top of the Internet that deliver content to users from the network edges, thus reducing response time compared to obtaining content directly from the origin server.

Problem 1: **Content Outsourcing**. Let there be a network topology X consisting of $|X|$ network nodes, and a set N of $|N|$ surrogate servers, whose locations are appropriately [29] selected upon the network topology. Also, let there be a set O of $|O|$ unique objects of the Web server content, which has an outsourcing contract with the CDN provider, where N_o denotes the set of locations where object o has been replicated and $l_{o(j)}$ denotes that the object o is located at the j-th surrogate. It has to be noticed here, that N_o includes also the origin Web server. For a set W of $|W|$ users, where O_w represents the set of objects requested by client w, we will denote the response time experienced by user i ($i \in W$), when s/he retrieves object o from the surrogate j($j \in N$), as $T^o_{i,j}$. Therefore, CDNs should adapt an *outsourcing policy* in order to select the content to be outsourced ($O^{outsourcingPolicy}_{outsourced} \subseteq O$) in such a way that is minimizes the following equation:

$$response\_time = \sum_{i \in W} \left(\sum_{o \in O_i} \left(\min_{j \in N_o} T^o_{i,l_{o(j)}} \right) \right) \quad (1)$$

subject to the constraint that the total replication cost is bounded by $OS \leq C(N)$, where OS is the total size of all the replicated objects to surrogate servers, and $C(N)$ is the cache capacity of the surrogate servers.

Regarding where to replicate the outsourced content, we propose a heuristic [8] replica placement policy [26]. According to this policy, the outsourced content is replicated to surrogate servers with respect to the total network's latency from the origin Web server to the surrogate servers, making a conservative assumption that the clients' request patterns are homogeneous. Specifically, for each outsourced object, we find which is the best surrogate server in order to place it (produces the minimum network latency). Then, we select from all the pairs of outsourced object-surrogate server that have been occurred previously, the one which produces the largest network latency, and thus place this object to that surrogate server. The above process is iterated until all the surrogate servers become full.

[8] Authors in [18] have shown that this problem is NP complete.

Variable	Description
$G(V, E)$	Web graph where V is the number of nodes and E is the number of edges
$U(V_u, E_u)$	subgraph of the graph G here V_u is the number of nodes and E_u is the number of edges
$M_{i,j}$	adjacency matrix of the graph G
d_v	graph's degree of node v
$d^{in}_v(U)$	number of edges connecting node v to other nodes belonging to U
$d^{out}_v(U)$	number of edges connecting node v to other nodes belonging to $G \cap U$
X	set of nodes in the internetwork topology
N	set of surrogate servers
$C(N)$	available storage capacity of N surrogate servers
S	total size of all replicated objects
W	set of users
O	set of unique objects of a Web site
O_w	set of objects requested by user u
$T^o_{W,N}$	the response time for an object o between the i-th user ($i \in W$) and node j ($j \in N$)
N_o	set of locations where object o has been replicated
$l_{o(j)}$	location of object o at the j-th surrogate server
C	set of the Web communities where $C = \{C_1, C_2, ...\} \subset G$
$sp_{st}(v)$	the number of the shortest paths from a node s to t that contains the node v
$BC(v)$	Betweenness Centrality of node v
$B[i, j]$	the number of edges from community C_i to community C_j
$D(C_i, C'_j, G)$	similarity distance measure
$K(n, C)$	the set of communities that the node n has been assigned
$O^{outsourcingPolicy}_{outsourced}$	the set of objects which is selected to be outsourced by an outsourcing policy

TABLE I
Paper's notation overview.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

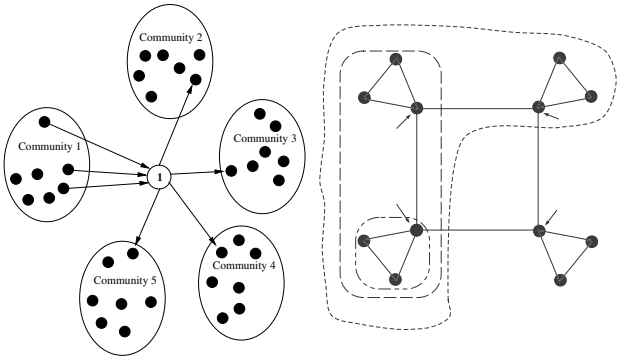


Fig. 2. Sample communities where "hard Web page community" approach is not appropriate for Content Outsourcing in CDNs.

B. Web page communities identification

The outsourced content is replicated to CDN's surrogate servers in the form of Web page communities. As it has been described in Section II, several approaches have been proposed in the literature [12], [13], [20] in order to identify Web page communities. However, all these approaches require heavy parameterization to work in CDNs' environment and present some important weaknesses, since we could end up either with a huge number of very small communities or with a single community comprised by the entire Web server content. On the other hand, max-flow communities [12] seem more appropriate, but they suffer from their strict definition (in the sequel we provide detailed explanation of this).

Formally, we consider a graph $G = (V, E)$, where V is a set of nodes and E is a set of edges. Let $d_i(G)$ be the degree of a generic node i of the considered graph G, which, in terms of the adjacency matrix M of the graph, is $d_i = \sum_{j,i \neq j} M_{i,j} + \sum_{j,i \neq j} M_{j,i}$. If we consider a subgraph $U \subset G$, to which node i belongs, we can split the total degree d_i in two quantities: $d_i(U) = d_i^{in}(U) + d_i^{out}(U)$. The first term of the summation denotes the number of edges connecting node i to other nodes belonging to U, i.e., $d_i^{in}(U) = \sum_{j \in U, i \neq j} M_{i,j}$. The second term of the summation formula denotes the number of connections of node i toward nodes in the rest of the graph ($G \cap U'$, where U' is the complement of graph U) i.e., $d_i^{out}(U) = \sum_{j \notin U, i \neq j} M_{j,i}$. Specifically, a Web page community has been defined in [12] as follows:

Definition 1: **The hard Web page community**. A subgraph $U(V_u, E_u)$ of a Web site graph G constitutes a Web page community, if every node v ($v \in V_u$) satisfies the following criterion:

$$d_v^{in}(U) > d_v^{out}(U) \tag{2}$$

This definition is quite restrictive for the CDNs' content outsourcing problem since it allows a node to be in at most one community or in no community at all. For instance, according to Definition 1, node 1 in the left part of Figure 2 will not be assigned to any community; also no community will be discovered for the graph shown in the right part of Figure 2, although at a first glance, we could recognize four communities based on our human perception, i.e., the four triangles; a careful look would puzzle us whether the four nodes (pointed to by arrows) at the corners of the square really belong to any community.

Thus, a new and more flexible definition of the community is needed, that: (a) will allow for overlapping communities, (b) will be based only on the hyperlink information, assuming that two pages are "similar"/"correlated" iff there exists a link between them, (c) will not make use of artificial "weights" on the edges of the Web site graph, and (d) will not make use of the direction of links in the Web site graph (the user browses the Web using both hyperlinks and the "back button").

Definition 2: **The Generalized Web page community [32]**. A subgraph U of a Web site graph G constitutes a Web page community, if it satisfies the following condition:

$$\sum_{v \in U} d_v^{in}(U) > \sum_{v \in U} d_v^{out}(U), \tag{3}$$

i.e., the sum of all degrees within the community U is larger than the sum of all degrees toward the rest of graph the G.

Obviously, every hard Web page community may also be a generalized Web page community, but the opposite is not always true. Thus, our definition overcomes the limitation explained in Figure 2 by implying more generic communities. Thus, rather than focusing on each individual member of the community, like the Definition 1, which is an extremely local consideration, we consider relative connectivity between the nodes. Certainly, this definition does not lead to unique communities, since by adding or deleting "selected" nodes the property may still hold. Based on the above definition, we regard the identification of generalized Web page communities as follows:

Problem 2: **Correlation Clustering Problem**. Given a graph with V vertices, find the (maximum number of) possibly overlapping groups of vertices (i.e., a non-crisp clustering [11]), such that the number of edges within clusters is maximized and the number of edges between clusters is minimized, such that Definition 2 holds for each group.

Finding the optimal correlated clusters is unfeasible, since this clustering problem is NP-hard [2]. To deal with this problem, we propose a heuristic algorithm, namely CiBC, which identifies generalized Web page communities.

IV. The CiBC algorithm

CiBC is a heuristic CDN's outsourcing policy which identifies generalized Web page communities from a Web server content. CiBC is a *hybrid* approach, since the identified communities are based on both local and global graph's properties (discussed in Subsection II-B). The Web server content is represented by a Web site graph $G = (V, E)$, where its nodes are the Web pages and the edges depict the hyperlinks among Web pages. Given the Web site graph (Web server content structure), CiBC outputs a set of Web page communities. These communities constitute the set of objects ($O_{outsourced}^{CiBC}$) which are outsourced to the surrogate servers.

CiBC consists of two phases. In the first phase, we compute the BC of the nodes of the Web site graph. In the second phase, the nodes of the Web site graph are accumulated around the nodes which have the lowest BC, called *pole-nodes*. Then,

the resulted groups are processed and a set of Web page communities is produced. The algorithm is given in Figure 3.

Algorithm 1 The CiBC algorithm

Input: $G(V, E)$: Web site graph;
Output: $C = \{C_1, C_2, ...\}$: A set of Web page communities
Phase I: Computation of Betweenness Centrality
1: **for all** $v \in G$ **do**
2: compute Betweenness Centrality (BC);
3: **end for**
4: sort V nodes of G by ascending BC;
Phase II: Accumulation around Pole-Nodes
/*l: the number of groups*/
5: l=1;
6: **repeat**
7: **for** $i = 1$ to V **do**
8: **if** $V(i) \in C$ **then**
9: continue;
10: **end if**
11: $C_l = C_l \bigcup \{V(i)\}$;
12: $C_l = C_l \bigcup$ {nodes directly connected with $V(i)$};
13: $C_l = C_l \bigcup$ Bounded-BFS(graph: G, depth: $\sqrt{V}$, starting node:$V(i)$);
14: $l = l + 1$;
15: **end for**
16: **for** $i = 1$ to l **do**
17: **for** $j = 1$ to l **do**
18: $B[i, j]$=number of edges from C_i to C_j;
19: $B[i, i]$=number of internal edges of C_i;
20: **end for**
21: **end for**
22: Find (C_x, C_y)=maximum($\frac{B[x,y]}{B[x,x]}$);
23: **if** (size of $C_x \cup C_y \leq$ cache size of surrogate) and ($B[x, y] \geq B[x, x]$) **then**
24: $C_x \leftarrow C_x \cup C_y$;
25: delete C_y;
26: $l = l - 1$;
27: **end if**
28: **for** $i = 1$ to l **do**
29: V_{C_i}=nodes in C_i;
/*Degenerate cases of communities are removed*/
30: **if** (number of $V_{C_i} \leq 3$) or (size of $V_{C_i} >$ cache size of surrogate) **then**
31: $V_{C_i} \rightarrow$ unassigned nodes;
32: delete C_i;
33: $l = l - 1$;
34: **end if**
35: **end for**
36: $C_{l+1} = \{V \notin C\}$;
37: **until** size of $C_{l+1} >$ cache size of surrogate

Fig. 3. The CiBC algorithm.

A. Phase I: Computation of Betweenness Centrality

Initially, the BC of nodes is computed [5] (line 2), providing information about how central a node is within the Web graph. Formally, the BC of a node can be defined in terms of a probability as follows:

Definition 3: **Betweenness Centrality**. Given a Web graph $G = (V, E)$, $sp_{st}(v)$ is the number of the shortest paths from a node $s \in V$ to $t \in V$ that contains the node $v \in V$ and sp_{st} is the number of shortest paths from s to t. The betweenness value of node v for pairs s, t ($b_{st}(v)$) is the probability that node v falls on a randomly selected shortest path connecting s with t. The overall BC of a node v is obtained by summing up its partial betweenness values for all unordered pairs of nodes $\{(s, t)|s, t \in V, s \neq t \neq v\}$:

$$BC(v) = \sum_{s \neq t \neq v \in V} \frac{sp_{st}(v)}{sp_{st}} \quad (4)$$

In our framework, the BC of a particular Web page captures the level of navigational relevance between this page and the other Web pages in the underlying Web server content. This relevance can be exploited to identify tightness and correlation between the Web pages, towards forming a community. In order to compute the BC, we use the algorithm described in [5], since to the best of our knowledge, it has the lowest computational complexity. Specifically, according to [5], its computation approach is $O(VE)$ where V is the number of nodes and E is the number of edges. Then, the nodes of the graph G is sorted by the ascending order of BC values (line 4).

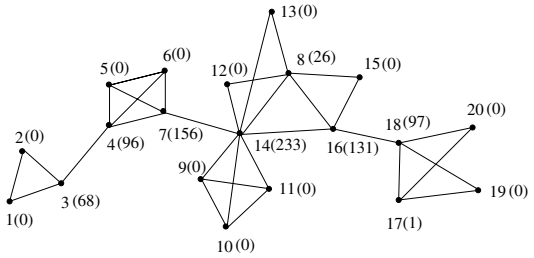


Fig. 4. Calculation of BC for a sample graph.

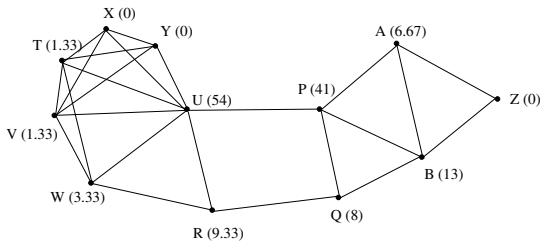


Fig. 5. Calculation of BC for another sample graph. The numbers in parentheses denote the BC index for the respective node ID.

B. Phase II: Accumulation around Pole-Nodes

The first step in this phase is to accumulate the nodes of the Web graph G around the pole-nodes. It is an iterative process where in each iteration we select the node with the lowest BC (called *pole-node*). Specifically, we start the formation of the communities from the pole-nodes since, as it is pointed by [22], the nodes with high BC have large fan-outs or are articulation points (nodes whose removal would increase the number of connected components in the graph) of the graph.

For instance, consider the Figure 4, where the communities of the graph sample are well formed and intuitive. The numbers in parentheses denote the BC index for the respective node ID (the number left to the parentheses). We observe that in some cases, the nodes with high BC are indeed central in the communities (e.g., node 8), but there are also cases, where the nodes with high BC are the nodes which are in-between the communities (e.g., nodes: 3, 4, 14, 18). Figure 5 depicts another example, where the communities are not obvious at all. If we start the formation of communities from the node with the highest BC, i.e., node U, then it is possible that we will end up with a single community covering the whole graph. Therefore, we accumulate the nodes of the Web graph around the ones which have low BC. This process comes to an end when all the nodes of the graph have been assigned to a group.

In this context, the selected pole-node is checked if it belongs to any group (lines 8). If not, it indicates a distinctive one (line 11) and then, it is expanded by the nodes which are directly connected with it (line 12). Afterwards, the resulted group is expanded by the nodes which are traversed using the bounded-BFS (Breadth First Search) algorithm [33] (line 13). A nice feature of the bounded-BFS algorithm is that the resulted group is uniformly expanded until a specific depth of the graph. A typical value for the graph's depth is $\sqrt{V}$ [2]. Then, the above procedure is repeated by selecting as pole-node the one with the next lowest BC. At the end, a set of groups would have been created.

The next step is to minimize the number of these groups by merging the identified groups so as to produce generalized Web page communities. Therefore, we define an $l \times l$ matrix B (l denotes the number of groups), where each element $B[i,j]$ corresponds to the number of edges from group C_i to group C_j. The diagonal elements $B[i,i]$ correspond to the number of internal edges of the group C_i ($B[i,i] = \sum_{v \in C_i} d_v^{in}(C_i)$). On the other hand, the sum of a row of matrix B is not equal to the total number of edges. For instance, if a node x belongs to communities C_i and C_j, and there is an edge $x \longrightarrow y$, then this edge counts once for $B[i,i]$ (since it is an internal edge), but it also counts for $B[i,j]$ (since y belongs also to C_j). As far as the merging of the groups is concerned, the CiBC consists of several iterations, where one merge per iteration takes place. The iterations come to an end (convergence criterion) when there is not any other pair to be merged. In each iteration, the pair of groups is selected for merging which has the maximum $\frac{B[i,j]}{B[i,i]}$ (line 24). We consider that a merge of the groups C_i, C_j takes place when the following condition is true:

$$B[i,j] \geq B[i,i] \tag{5}$$

The above equation satisfies Definition 2, since the sum of the internal edges of the resulted communities would be larger than the sum of edges toward to the rest of the graph.

Degenerate cases of communities (i.e., communities with only two or three nodes or communities which are larger than the cache size) are removed (line 30). The algorithm terminates when there is no change in the number of communities.

C. Time and space complexity

The CiBC's first phase is an initialization process, that calculates the BC of nodes. This computation can be done in time complexity $O(VE)$ for a graph $G(V,E)$ [5]. Considering the sparse nature of the Web site graphs, i.e., $|E| = O(|V|)$, it is obvious that this computation is very fast and it does not affect the performance of the algorithm. Therefore, even for the case of dynamic graphs where we have to execute it periodically, the overhead due to this computation is negligible.

The performance of the second phase of CiBC is highly depended on the maximum number of iterations executed by CiBC, which is proportional to the initial number of the selected pole-nodes. Considering that a typical value of maximum number of nodes for each community is $m = \sqrt{V}$ [2], the time complexity with respect to the number of nodes is $O(V\sqrt{V})$, since in each iteration, all the pairs of the communities are checked. In particular, the time that is required is $m^2 + (m-1)^2 + \ldots = \sum_{n=0}^{m}(m-n)^2 \cong m^3 \sum_{n=0}^{m}(1-\frac{n}{m})^2$, which means that the complexity is $O(m^3)$. As far as the memory space is concerned, the space complexity is $O(V)$ (i.e., an $m \times m$ matrix).

To sum up, the time and space computational requirements of CiBC algorithm are mainly dependent on the number of graph's nodes. On the other hand, its performance is only linearly affected by the number of edges.

V. Simulation testbed

The CDNs' providers are real time applications and they are not used for research purposes. Therefore, for the evaluation purposes it is crucial to have a simulation testbed [4] for the CDN's functionalities and the Internet topology. Furthermore, we need a collection of Web users' traces which access a Web server content through a CDN, as well as, the topology of this Web server content (in order to identify the Web page communities). Although we can find several users' traces on the Web[9], real traces from CDN's providers are not available. Thus, we are faced to use artificial data. Moreover, using artificial data enables us to validate extensively the proposed approach. In this framework, we have developed a full simulation environment, which includes the following: a) a system model simulating the CDN infrastructure, b) a Web server content generator, modelling file sizes, linkage, etc., c) a client request stream generator capturing the main characteristics of Web users' behavior, and d) a network topology generator. Table II presents the default parameters for the setup.

A. CDN model

To evaluate the performance of the proposed algorithm, we used our complete simulation environment, called CDNsim, which simulates a main CDN infrastructure and is implemented in the C programming language. A demo can be found at http://oswinds.csd.auth.gr/~cdnsim/. It is based on the OMNeT++ library[10] which provides a discrete event simulation environment. All CDN networking issues, like

[9]Web traces: http://ita.ee.lbl.gov/html/traces.html

[10]http://www.omnetpp.org/article.php?story=20080208111358100

Parameter	Description	Default Value
Number of surrogate servers	Randomly placed	100 servers
Cache Size	Surrogate Server's Cache Size	40% of the Web server content's size
Network Topology	AS Internet Topology	3037 nodes
Web server content	Produced by FWGen	16000 nodes, 121000 edges; total size: 4.2 GB
Number of Clients	Randomly placed to network topology	39847 clients
Number of Requests	Produced by a requests' generator [21]	1 million requests

TABLE II
SIMULATION TESTBED.

surrogate server selection, propagation, queueing, bottlenecks and processing delays are computed dynamically via CDNsim, which provides a detailed implementation of the TCP/IP protocol (and HTTP 1.1), implementing packet switching, packet re-transmission upon misses, objects' freshness etc.

By default, CDNsim simulates a cooperative push-based CDN infrastructure [31], where each surrogate server has knowledge about what content (which has been pro-actively pushed to surrogate servers) is cached to all the other surrogate servers. If a user's request is missed on a surrogate server, then it is served by another surrogate server. In this framework, the CDNsim simulates a CDN with 100 surrogate servers which have been located all over the world. The default size of each surrogate server has been defined as the 40% of the total bytes of the Web server content. Each surrogate server in CDNsim is configured to support 1000 simultaneous connections. Finally, the CDN's surrogate servers do not apply any cache replacement policy. Thus, when a request cannot be served by the CDN's infrastructure, it is served by the origin server without being replicated by any CDN's surrogate server. The outsourced content has a priori been replicated to surrogate servers using the heuristic approach that has been described in subsection III-A. This heuristic policy is selected as the default replica placement one since it achieves the highest performance for all the outsourcing policies.

B. Web server content generation

In order to generate the synthetic Web site graphs we developed a tool, the so-called the FWGen[11]. The FWGen tool produces synthetic but realistic Web graphs since it encapsulates many of the patterns found in real world graphs, including power-law and log-normal degree distributions, small diameter and communities "effects". The parameters that are given to FWGen are: (a) number of nodes, (b) number of edges or density related to the respective fully connected graph, (c) number of communities to generate, (d) skew, which reflects the relative sizes of the generated communities, and finally (e) the assortativity factor, which gives the percentage of the edges that are intra-community edges. The higher the assortativity is, the stronger communities are produced. For instance, if the assortativity is 100%, then the generated communities are disconnected with each other. In general, research has shown that the assortativity factor of a Web site's graph is usually high [22]. From a technical point of view, FWGen creates two files. The first one is the graph and the second one records the produced communities. Thus, we can compare the communities identified by the CiBC using a similarity distance measure. For our experiments, the produced Web graphs have 16000 nodes and their sizes are 4.2 GB. The number of edges slightly varies (about 121000) with respect to the values of the assortativity factor.

[11]FWGen Tool: http://delab.csd.auth.gr/∼dimitris

C. Requests' generation & network topology

As far as the requests' stream generation is concerned, we used the generator described in [21], which reflects quite well the real users' access patterns. Specifically, this generator, given a Web site graph, generates transactions as sequences of page traversals (random walks) upon the site graph, by modelling the Zipfian distribution to pages. In this work, we have generated 1 million users' requests. Each request is for a single object, unless it contains "embedded" objects. We consider that the requests arrive according to a Poisson distribution with rate equal to 30. Then, the Web users' requests are assigned to CDN's surrogate servers taking into account the network proximity and the surrogate servers' load, which is the typical way followed by CDNs' providers (e.g., Akamai) [36]. Some more technical details about how CDNsim manages the users' requests can be found in [32] and at http://oswinds.csd.auth.gr/∼cdnsim/. Finally, concerning the network topology, we used an AS-level Internet topology with a total of 3037 nodes. This topology captures a realistic Internet topology by using BGP routing data collected from a set of 7 geographically-dispersed BGP peers.

VI. EXPERIMENTATION

A. Examined policies

In order to evaluate our proposed algorithm, we conducted extensive experiments comparing CiBC with another state-of-the-art policy. It should be noticed that although there are several approaches which deal with finding communities (dense clusters) in a graph, the CPM algorithm has been preferred to be compared with CiBC. The reason is that CPM, in contrast to the most common approaches (flow-based, matrix-based such as SVD, distance-based etc), may produce in an effective way overlapped Web page communities which are in accordance with our target application. Furthermore, we investigate the performance of our earlier proposal, i.e., C3i, and of a Web caching scheme.

- **Clique Percolation Method (CPM):** The outsourced objects obtained by the CPM ($O^{CPM}_{outsourced}$) correspond to k-clique percolation clusters in the network. The k-cliques are complete sub-graphs of size k (in which each node is connected to every other nodes). A k-clique percolation cluster is a sub-graph containing k-cliques that can all reach each other through chains of k-clique adjacency, where two k-cliques are said to be adjacent if they share $k-1$ nodes. Experiments have shown that this method is very efficient when it is applied

on large graphs [28]. In this framework, the communities are extracted with the CPM using the CFinder package, which is freely available from http://www.cfinder.org.

- **Correlation Clustering Communities identification (C3i):** The outsourced communities obtained by the C3i are identified by naively exploring the structure of the Web site [32]. This results in requiring parameterization to identify communities. It is an algorithm that came out from our earlier preliminary efforts to develop a communities-based outsourcing policy for CDNs.
- **Web caching scheme (LRU):** The objects are stored reactively to proxy cache servers. We consider that each proxy cache server follows the LRU (Least Recently Used) replacement policy since it is the typical case for the popularity of proxy servers (e.g., Squid).
- **No Replication (W/O CDN):** All the objects are placed on the origin server and there is no CDN/no proxy servers. This policy represents the "worst-case" scenario.
- **Full Replication (FR):** All the objects have been outsourced to all the CDN's surrogate servers. This (unrealistic) policy does not take into account the cache size limitation. It is, simply, the optimal policy.

B. Evaluation measures

We evaluate the performance of the above methods under a regular traffic and under a flash crowd event (a large amount of requests is served simultaneously). It should be noted that for all the experiments we have a warm-up phase for the surrogate servers' caches. The purpose of the warm-up phase is to allow the surrogate servers' caches to reach some level of stability and it is not evaluated. The measures used in the experiments are considered to be the most indicative ones for performance evaluation. Specifically, the following measures are used:

- **Mean Response Time (MRT):** the expected time for a request to be satisfied. It is the summation of all requests' times divided by their quantity. This measure expresses the users' waiting time in order to serve their requests. The overall response time consists of many components, namely DNS delay, TCP setup delay, network delay between the user and the server, object transmission delay and so on. Our response time definition implies the total delay due to all the aforementioned components. In the experiments for regular traffic, the results depict the total response time, since DNS and TCP setup delay are negligible, whereas the network delay dominates the object transmission delay. So it makes no sense to present individual figures for these components, and we resort to the total response time as the measure of performance for regular traffic. In the experiments simulating a flash crowd, we provide more detailed results breaking down the response time to its components.
- **Response time CDF:** the Cumulative Distribution Function (CDF) here denotes the probability of having response times lower or equal to a given response time. The goal of a CDN is to increase the probability of having response times around the lower bound of response times.
- **Replica Factor (RF):** the percentage of the number of replica objects to the whole CDN infrastructure with respect to the total outsourced objects. It is an indication about the cost of maintaining the replicas "fresh". High values of replica factor mean high data redundancy and waste of money for the CDNs' providers.
- **Byte Hit Ratio (BHR):** It is defined as the fraction of the total number of bytes that were requested and existed in the cache of the closest to the clients surrogate server to the number of bytes that were requested. A high byte hit ratio improves the network performance (i.e., bandwidth savings, low congestion).

In general, replication and response time are interrelated and the pattern of dependence among them follows the rule that the increased replication results in reduced response times, because the popular data are outsourced closer to the final consumers. Specifically, the lowest mean response time (upper limit), which can be achieved by an outsourcing policy, is observed when its replica factor is 1. In such a case, all the outsourced objects ($O_{outsourced}^{outsourcingPolicy}$) of the underlying outsourcing policy have been replicated by all surrogate servers.

In addition, the performance of the examined outsourced policies is evaluated. Specifically, CiBC approach is compared with CPM and C3i with respect to the following measures in order to assess their speed (how quickly they provide a solution to the outsourcing problem?) as well as their quality (how well they identify the communities?):

- **CPU Time:** to measure the speed of the examining algorithms, since it is a value reported by using the system call time() of the Unix kernel and conveys the time that the process remained in the CPU.
- **Similarity distance measure:** to measure the degree of discrepancy between identified and real communities (the produced communities by the FWGen tool). For this purpose, a similarity distance measure is used, which measures the mean value of the difference of the objects' correlation within two Web page communities sets (identified communities by an outsourcing policy and FWGen's communities). Specifically, the distance between C_i and C_i' is defined as

$$D(C_i, C_i', G) = \frac{\sum_{n_1 \in V} \sum_{n_2 \in V} |Y - Z|}{|V|(|V| - 1)}, \qquad (6)$$

$Y = \frac{\sum_{c \in K(n_1, C_i)} N(n_2, c)}{|K(n_1, C_i)|}$, $Z = \frac{\sum_{c \in K(n_1, C_i')} N(n_2, c)}{|K(n_1, C_i')|}$, and $K(n, C)$ gives the set of communities ($[1 \dots |C|]$) that the node n has been assigned to, and N(n,c)=1 if $n \in c$ and 0 otherwise. According to the above, if C_i and C_i' are equal then $D = 0$.

Finally, we use the *t-test* to assess the reliability of the experiments. T-test is a significance test that can measure results effectiveness. In particular, the t-test would provide us an evidence whether the observed mean response time is due to chance. In this framework, the *t* statistic is used to test the following null hypothesis (H_0):

H_0: The observed mean response time and the expected mean response time are significantly different.

The t statistic is computed by the following formula:

$$t = \frac{\overline{x} - \mu}{s_{\overline{x}}} \qquad (7)$$

where $\overline{x}$ and μ are the observed and expected mean response times respectively. The variable $s_{\overline{x}}$ is the standard error of the mean. Specifically, $s_{\overline{x}} = \frac{s}{\nu}$, where $s = \sqrt{\frac{\sum_{i=1}^{\nu}(x_i - \overline{x})}{\nu - 1}}$, and ν is the number of observations in the sample. A small value of the t statistic shows that the expected and the actual values are not significantly different. In order to judge whether t statistic is really small, we need to know a critical value ($t_{df;\alpha}$) for the boundary of the area of hypothesis's rejection. For finding this critical value, we should define the level of significance α (probability of erroneously rejecting the null hypothesis) and the degrees of freedom ($df = \nu - 1$). Thus, if the value of t statistic, computed from our data, is smaller than $t_{df;\alpha}$, we reject the null hypothesis at the level of significance α. In such a case, the probability that the observed mean response time is due to chance is less than α.

VII. Evaluation

A. Evaluation under regular traffic

1) Client-side evaluation: Firstly, we tested the competing algorithms with respect to the average response time, with varying the assortativity factor. The results are reported in Figure 6. The y-axis represents time units according to OMNeT++'s internal clock and not some physical time quantity, like seconds, minutes. So the results should be interpreted by comparing the relative performance of the algorithms. This means that if one technique gets a response time 0.5 and some other gets 1.0, then in the real world the second one would be twice as slow as the first technique. The numbers which are included within the parentheses represent the number of communities identified by the underlying algorithm for different values of the assortativity factor.

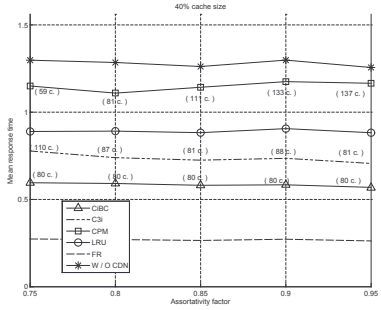


Fig. 6. Mean response time vs. assortativity factor.

We observe that the CiBC achieves the best performance compared to CPM, LRU, C3i. Specifically, the CiBC improves the mean response time at about 60% w.r.t. CPM, about 38% w.r.t. LRU and around 20% w.r.t. C3i for regular traffic conditions. In general, the communities-based outsourcing approach is advantageous when compared to LRU and W/O CDN, only under the premise that the community-identification algorithm does a good job in finding communities. Although it is impossible to check manually the output of each algorithm so as to have an accurate impression of all the identified communities, we observe that CPM achieves its lowest response time when the assortativity factor is equal to 0.8. In this case it finds 81 communities, a number that is very close to the number of communities found by CiBC and C3i algorithms. Presumably, it does not discover exactly the same communities and this is the reason for achieving higher response time than CiBC and C3i. In general, the low performance of CPM is attributed to the fact that it fails to identify large-scale communities, which makes this algorithm worse than LRU. This observation strengthens our motive to investigate efficient and effective algorithms for community discovery, because they have dramatic impact upon the response time.

As far as the performance of FR and W/O CDN is concerned, since these approaches represent the unfeasible lower-bound and impractical upper-bound of the mean response time respectively, their performance is practically constant; they are not affected by the assortativity factor. They do not present any visible up/down trend, and any observable variations are due to the graph and request generation procedure. This is a natural consequence, if we consider that they perform no clustering.

The general trend for the CiBC algorithm is that the response time is slightly decreasing with larger values of assortativity factor. The explanation for this decrease is that, due to the request stream generation method, which simulates a random surfer, as the value of assortativity factor grows (more dense communities are created), the random surfers browse mainly within the same communities. Figure 7 depicts the CDF for a Web graph with assortativity factor 0.75. The y-axis presents the probability of having response times lower or equal to a given response time. From this figure, we see that the CiBC serves a large volume of requests in low response times. Quite similar results have been observed for the other values of assortativity factor.

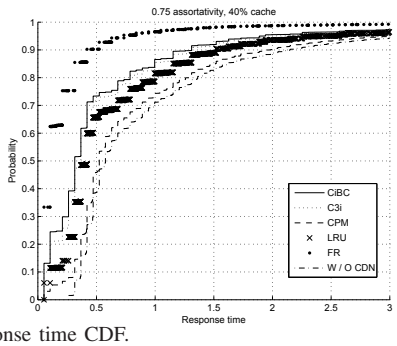


Fig. 7. Response time CDF.

Figure 8 presents the BHR of the examined policies for different values of the assortativity factor. The CiBC has the highest BHR, around 40%. This is a remarkably significant improvement, if we consider that the BHR of a typical proxy cache server (e.g., Squid) is about 20% [35], and it is attributed to the cooperation among the surrogate servers. On the other hand, the CPM and LRU present low BHR, without exceeding 20%. The increase of the BHR is a very significant matter, and even the slightest increase can have very beneficial results to the network, since fewer bytes are travelling in the net, and thus the network experiences lower congestion. In our case, we see that although CiBC is only 6% (on the average) better than C3i in terms of BHR, this gain is translated into 20% reduction of the average response time (Figure 6).

Overall, the experimentation results showed that the com-

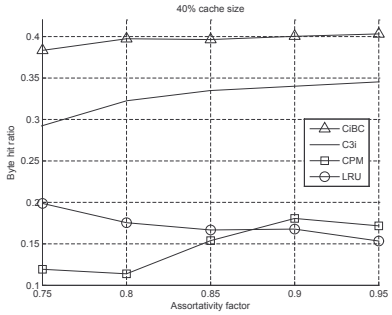


Fig. 8. BHR under regular traffic.

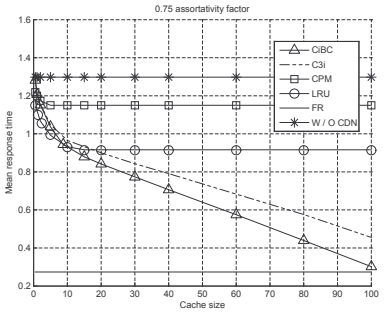


Fig. 9. Mean response time vs. cache size.

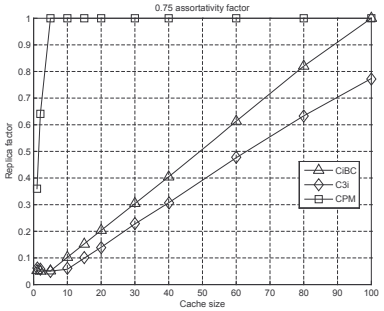


Fig. 10. Replica factor vs. cache size.

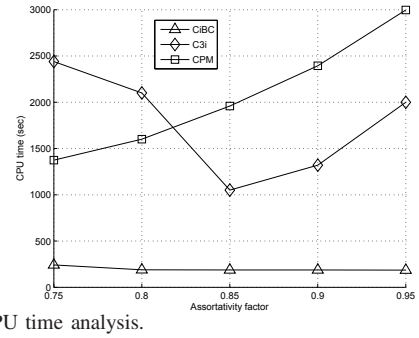


Fig. 11. CPU time analysis.

munities-based approach is a beneficial one for CDN's outsourcing problem. Regarding the CiBC's performance, we observed that the CiBC algorithm is suitable to our target application, since it takes into account the CDNs' infrastructure and theirs challenges, reducing the users' waiting time and improving the QoS of the Web servers content.

2) Server-side Evaluation: The results are reported in Figures 9 and 10, where x-axis represents the different values of cache size as a percentage of the total size of distinct objects. A percentage of cache size equal to 100% may not imply that all objects are found in the cache due to the overlap among the outsourced communities. In general, an increase in cache size results in a linear (for caches larger than 15%) decrease in mean response time and in a linear (for caches larger than 10%) increase in the replica factor. CiBC achieves low data redundancy as well as low mean response time. For small cache sizes ($<$ 10% of total objects' volume) it is outperformed by LRU. This is due to the fact that there is a time penalty to bring the missing objects no matter how the cache size is increased. However, as the cache size increases, the CiBC outperforms LRU.

On the other hand, the CPM algorithm presents high data redundancy, without exhibiting the smooth behavior of CiBC. Its replica factor is increased almost instantly by increasing the cache size, with no significant improvement in mean response time. Specifically, the CPM identifies Web page communities where each one has a small number of objects. Thus, the CPM reaches the maximum value of replica factor for small cache sizes. This drawback is not exhibited by C3i, which follows the CiBC's pattern of performance, although it lags w.r.t. CiBC at a percentage which varies from 10% to 50%. Finally, the unfeasible FR approach, as expected, presents high data redundancy, since all the outsourced objects have been replicated to all the surrogate servers.

CiBC achieves low replica factor. This is very important for CDNs' providers since a low replica factor reduces the computing and network resources required to remain the content updated. Furthermore, a low replica factor reduces the bandwidth requirements for Web servers content, which is important economically for both individual Web servers content and for the CDN's provider itself [4].

Figure 11 depicts the performance of the examining algorithms with respect to run times for various values of the assortativity factor. The efficiency of the CiBC is obvious since it presents the lowest run times for all the values of the assortativity factor that have been examined. Specifically, CiBC is 6–16 times faster than CPM. Furthermore, we observe that the speed performance of CiBC is independent of the assortativity factor, whereas the CPU time for CPM depends on this factor at an exponential fashion. The behavior of C3i depends a lot on the assortativity factor; when the communities are "blurred" (i.e., low assortativity) its execution time is two times that of CPM, and only when the communities are more clearly present in the Web site graph, it becomes faster than CPM. Still, in the best case, C3i is three times slowest than CiBC, because the way it selects the kernel nodes [32] has a huge influence on the execution time. Even for high network assortativity, i.e., well-formed communities, an unfortunate selection of the kernel nodes might cause larger execution time when compared with the cases of networks with lower assortativity. Once again, this observation confirms the motive of this work to develop efficient and effective algorithms for community discovery in the context of content outsourcing.

3) Web Page Communities Validation: Figure 12 presents the validity of the examined outsourcing algorithms by computing the similarity distance measure (equation 6) between

the resulted communities (identified by CiBC, C3i and CPM) and the ones produced by the FWGen tool. The general observation is that when the communities are not "blurred" (i.e., assortativity factor larger than 0.75), CiBC manages to identify exactly the same number of communities (similarity distance≅0), although it is not guided by preset parameters used in other types of graph clustering algorithms. On the other hand, the communities identified by CPM are quite different than those produced. Finally, although C3i exhibits better performance than CPM, its performance approaches than of CiBC only for very large values of the network assortativity.

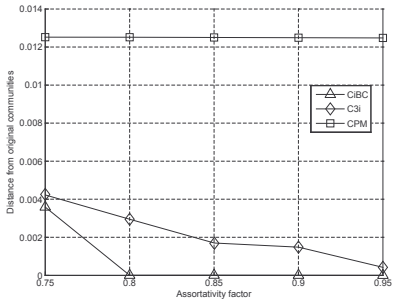


Fig. 12. Similarity distance measure.

4) Statistical Test Analysis: When a simulation testbed is used for performance evaluation, it is critical to provide evidence that the observed difference in effectiveness is not due to chance. Here, we conducted a series of experiments, making random permutation of users (keeping the rest of the parameters unchanged). Based on the equation 7, the value of t statistic is $t = -6.645$. The critical value of t-test is $t_{df;\alpha} = 3.29$ for level of significance $\alpha = 0.001$ and degrees of freedom $df = 175$. From the above it occurs that $t_{175;0.001} > t$, which means that the null hypothesis H_0 is rejected. Furthermore, we computed the $p - value$ of the test, which is the probability of getting a value of the test statistic as extreme as or more extreme than that observed by chance alone, if the null hypothesis H_0, is true. The smaller the $p - value$ is, the more convincing is the rejection of the null hypothesis. Specifically, we found that $p - value < 0.001$, which provides a strong evidence that the observed mean response time is not due to chance.

B. Evaluation under flash crowd event

Flash crowd events occur quite often and present significant problems to Web server content owners [17]. For instance, in commercial Web servers content, a flash crowd can lead to severe financial losses, as clients often resign from purchasing the goods and search for another, more accessible Web server content. Here, we investigate the performance of the algorithms under a flash crowd event. Recall that each surrogate server is configured to support 1000 simultaneous connections.

The experiments for the regular traffic measured the overall mean response time without breaking it into its components, since the network delay dominated all other components. The situation during a flash crowd is quite different; TCP setup delay is expected to contribute significantly to the MRT, or even to dominate over the rest of the delay components.

	Epoch-1	**Epoch-2**	**Epoch-3**
CiBC			
Mean TCP setup	0.00020	0.095	0.00021
Mean (Transmission+NetworkDelay)	0.00550	0.088	0.00568
C3i			
Mean TCP setup	0.00021	0.904	0.00024
Mean (Transmission+NetworkDelay)	0.00590	1.012	0.00684
LRU			
Mean TCP setup	0.00024	31.2540	0.00026
Mean (Transmission+NetworkDelay)	0.00351	33.3526	0.00410
CPM			
Mean TCP setup	0.00023	20.1432	0.00025
Mean (Transmission+NetworkDelay)	0.00836	26.8669	0.00855

TABLE III
RESPONSE TIME BREAKDOWN DURING A FLASH CROWD EVENT.

To understand the effect of a flash crowd on the performance of the algorithms, we simulated a situation where the users' request streams are conceptually divided into three equi-sized epochs. The first epoch simulates a situation with regular traffic, where the requests arrive according to a Poisson distribution with rate equal to 30. The second epoch simulates the flash crowd (i.e., the request rate increases two orders of magnitude and bursts of users' requests access a small number of Web pages), and finally, in the third epoch, the system returns again to the regular traffic conditions.

The dominant components of the response time are depicted in Table III (averages of values). DNS delay is excluded from this table because it was infinitesimal. The variance of these measurements was very large, which implied that the mean value would fail to represent the exact distribution of values. Thus, we plotted for each algorithm the total response time for each request in the course of the three epochs. The results are reported in the four graphs of Figure 13. It has to be noticed here, that we do not present results for the performance of the W/O CDN approach, because the Web server content collapsed quickly and did not serve any requests.

At a first glance, the results confirm the plain intuition that during a flash crowd event the response times increase dramatically; similar increase is observed in the variance of the response time. Looking at Table III, we observe that during regular traffic conditions (epoch1) TCP setup delay is one order of magnitude smaller than network and transmission delay. But during the flash crowd it becomes comparable or even exceeds the value of the other two components for the cases of CiBC and C3i. The flash crowd event has caused the other two algorithms to crash; their resources could cope with the surge of requests only for a small period during the flash crowd. This interval is very short for LRU; it is well known that the Web caching techniques cannot manage effectively the flash crowd events [17]. On the other hand, initially, CPM responds gracefully to the surge of requests (similar to the other two community-based outsourcing algorithms), but soon its lower-quality decisions w.r.t. to the identified communities, make it also to crash.

Turning to the two robust algorithms, namely CiBC and

	LRU	CiBC	C3i	W/O CDN	FR
MRT	0.29	0.23	0.27	0.42	0.11
BHR	0.15	0.42	0.33	N/A	1
RF	N/A	0.41	0.45	N/A	1

TABLE IV
PERFORMANCE OF THE ALGORITHMS FOR REAL INTERNET CONTENT.

C3i, we observe they are able to deal with the flash crowd. There are still many requests that experience significant delay, but the majority are served in short times. Apparently, CiBC's performance is superior than that of C3i. Moreover, as soon as the surge of requests disappears, the response times of CiBC are restored to the values it had before the flash crowd. C3i does not exhibit this nice behavior, which is attributed to its community-identification decisions.

In summary, the experiments showed that CiBC algorithm presents a beneficial performance under flash crowd events, fortifying the performance of Web servers content. Specifically, CiBC is suitable for a CDN's provider, since it achieves low data redundancy, high BHR and mean response times and it is robust in flash crow events.

C. Evaluation with real Internet content

To strengthen the credibility of the results obtained with the synthetic data, we evaluated the competing methods with real Internet content. The real Web site we used is the cs.princeton.edu. This Web site consists of 128805 nodes and 585055 edges. As far as the requests' stream generation is concerned, we used the same generator that we used for the synthetic data. Then, we "fed" these requests into CDNsim in exactly the same way we did for synthetic data. The execution of CiBC revealed 291 communities, whereas the execution of C3i revealed 1353 communities. As expected, due to its exponential time complexity, CPM did not run to completion and we terminated it after running for almost 5 days. The execution time for CiBC and C3i was 7963 and 19342 secs, respectively, on a Pentium IV 3.2GHz processor.

Table IV presents the relative performance results for the competing algorithms for a particular setting (i.e., 40% cache size), since the relative performance of the algorithms was identical to that obtained for the synthetic data. Notice that the replica factor is not measured for LRU, since it refers to static replication and not to "dynamic" replication. Similarly, BHR and RF have no meaning in the context of the W/O CDN approach. These results are in accordance with those obtained for the synthetic ones; indicatively, CiBC presents 27% higher BHR, 17% better response time and 10% lower replication than C3i.

VIII. CONCLUSION

In this paper, we dealt with the content selection problem; which content should be outsourced in CDN's surrogate servers. Differently from all other relevant approaches, we refrained from using any request statistics in determining the outsourced content, since we can not always gather them quite fast to address "flash crowds". We exploited the plain

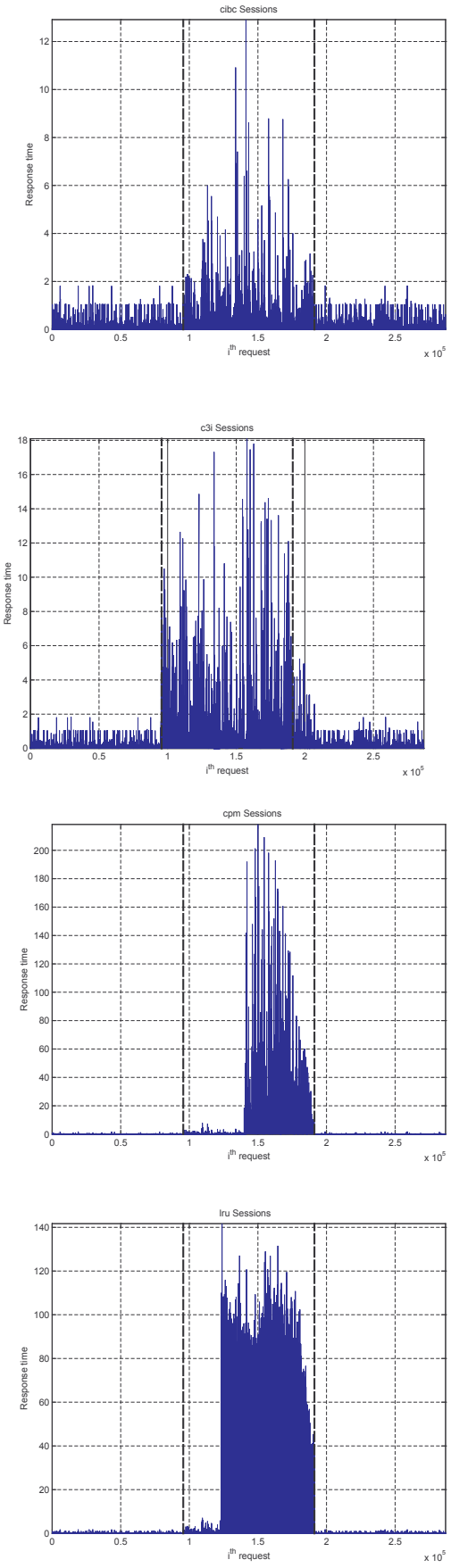


Fig. 13. Response time for CiBC, C3i, CPM and LRU (top to bottom).

assumption that Web page communities do exist within Web servers content, which act as attractors for the users, due to their dense linkage and the fact that they deal with coherent topics. We provided a new algorithm, called CiBC, to detect such communities; this algorithm is based on a new, quantitative definition of the community structure. We made them the basic outsourcing unit, thus providing the first "predictive" use of communities, which so far have been used for descriptive purposes. To gain a basic, though solid understanding of our proposed method's strengths, we implemented a detailed simulation environment, which gives Web server content developers more insight into how their site performs and interacts with advanced content delivery mechanisms. The results are very promising and the CiBC's behavior is in accordance with the central idea of the cooperative push-based architecture [31]. Using both synthetic and real data we showed that CiBC provides significant latency savings, keeping the replication redundancy at very low levels. Finally, we observed that the CiBC fortifies the CDN's performance under flash crowd events, providing significant latency savings.

Acknowledgments

The authors appreciate and thank the anonymous reviewers for their valuable comments and suggestions, which have considerably contributed in improving the papers content, organization and readability.

REFERENCES

[1] R. Andersen and K. Lang. Communities from seed sets. In *Proceedings of the ACM International Conference on World Wide Web (WWW)*, pages 223–232, 2006.

[2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, 2004.

[3] J. Fritz Barnes and R. Pandey. CacheL: Language support for customizable caching policies. In *Proceedings of the 4th International Web Caching Seminar*, 1999.

[4] L. Bent, M. Rabinovich, G. M. Voelker, and Z. Xiao. Characterization of a large Web site population with implications for content delivery. *World Wide Web Journal*, 9(4):505–536, 2006.

[5] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[6] A. Z. Broder, R. Lempel, F. Maghoul, and J. O. Pedersen. Efficient PageRank approximation via graph aggregation. *Information Retrieval*, 9(2):123–138, 2006.

[7] R. Buyya, A. K. Pathan, J. Broberg, and Z. Tari. A case for peering of content delivery networks. *IEEE Distributed Systems Online*, 7(10), 2006.

[8] S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's link structure. *IEEE Computer*, 32(8):60–67, 1999.

[9] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz. Efficient and adaptive Web replication using content clustering. *IEEE Journal on Selected Areas in Communications*, 21(6):979–994, 2003.

[10] A. Davis, J. Parikh, and W. E. Weihl. Edgecomputing: Extending enterprise applications to the edge of the Internet. In *Proceedings of the ACM International Conference on World Wide Web (WWW)*, pages 180–187, 2004.

[11] V. Estivill-Castro and J. Yang. Non-crisp clustering by fast, convergent, and robust algorithms. In *Proceedings of the International Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 103–114, 2001.

[12] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of Web communities. *IEEE Computer*, 35(3):66–71, 2002.

[13] G. W. Flake, K. Tsioutsiouliklis, and L. Zhukov. Methods for mining Web communities: Bibliometric, spectral, and flow. In A. Poulovassilis and M. Levene, editors, *Web Dynamics — Adapting to Change in Content, Size, Topology and Use*, pages 45–68. Springer, 2004.

[14] N. Fujita, Y. Ishikawa, A. Iwata, and R. Izmailov. Coarse-grain replica management strategies for dynamic replication of Web contents. *Computer Networks*, 45(1):19–34, 2004.

[15] K. Hosanagar, R. Krishnan, M. Smith, and J. Chuang. Optimal pricing of content delivery network services. In *Proceedings of the IEEE Hawaii International Conference on System Sciences (HICSS)*, volume 7, 2004.

[16] H. Ino, M. Kudo, and A. Nakamura. A comparative study of algorithms for finding Web communities. *Proceedings of the IEEE International Conference on Data Engineering Workshops (ICDEW)*, 2005.

[17] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In *Proceedings of the ACM International Conference on World Wide Web (WWW)*, pages 293–304, 2002.

[18] J. Kangasharju, J. Roberts, and K. W. Ross. Object replication strategies in Content Distribution Networks. *Computer Communications*, 25(4):367–383, 2002.

[19] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[20] T. Masada, A. Takasu, and J. Adachi. Web page grouping based on parameterized connectivity. In *Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 374–380, 2004.

[21] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. A data mining algorithm for generalized Web prefetching. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1155–1169, 2003.

[22] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[23] V. N. Padmanabhan and L. Qiu. The content and access dynamics of a busy Web site: Findings and implications. In *Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 111–123, 2000.

[24] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[25] G. Palla, A.-L. Barabasi and T. Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, 2007.

[26] G. Pallis, K. Stamos, A. Vakali, D. Katsaros, A. Sidiropoulos, and Y. Manolopoulos. Replication based on objects load under a content distribution network. In *Proceedings of the IEEE International Workshop on Challenges in Web Information Retrieval and Integration (WIRI)*, 2006.

[27] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006.

[28] P. Pollner, G. Palla, and T. Vicsek. Preferential attachment of communities: The same principle, but a higher level. *Europhysics Letters*, 73(3):478–484, 2006.

[29] L. Qiu, V. N. Padmnanabhan, and G. M. Voelker. On the placement of Web server replicas. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, volume 3, pages 1587–1596, 2001.

[30] M. Rabinovich and O. Spatscheck. *Web Caching and Replication*. Addison Wesley, 2002.

[31] M. Schiely, L. Renfer, and P. Felber. Self-organization in cooperative content distribution networks. In *Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA)*, pages, 109–118, 2005.

[32] A. Sidiropoulos, G. Pallis, D. Katsaros, K. Stamos, A. Vakali, and Y. Manolopoulos. Prefetching in content distribution networks via Web communities identification and outsourcing. *World Wide Web Journal*, 11(1):39–70, 2008.

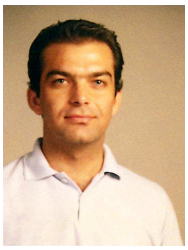
[33] H. Sivaraj and G. Gopalakrishnan. Random walk based heuristic algorithms for distributed memory model checking. *Electronic Notes in Theoretical Computer Science*, 89(1), 2003.

[34] S. Sivasubramanian, G. Pierre, M. van Steen, and G. Alonso. Analysis of caching and replication strategies for Web applications. *IEEE Internet Computing*, 11(1):60–66, 2007.

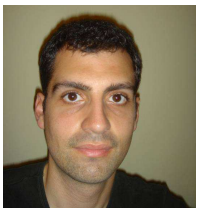
[35] A. Vakali. Proxy cache replacement algorithms: A history-based approach. *World Wide Web Journal*, 4(4):277–298, 2001.

[36] A. Vakali and G. Pallis. Content delivery networks: Status and trends. *IEEE Internet Computing*, 7(6):68–74, 2003.

[37] B. Wu and A. D. Kshemkalyani. Objective-optimal algorithms for long-term Web prefetching. *IEEE Transactions on Computers*, 55(1):2–17, 2006.



Dimitrios Katsaros was born in Thetidio (Farsala), Greece in 1974. He received a BSc in Computer Science from the Aristotle University of Thessaloniki, Greece (1997) and a Ph.D. from the same department on May 2004. He spent a year (July 1997-June 1998) as a visiting researcher at the Department of Pure and Applied Mathematics at the University of L'Aquila, Italy. Currently, he is a lecturer at the Department of Computer and Communication Engineering of University of Thessaly (Volos, Greece). He is editor of the book "Wireless Information Highways" (2005), co-guest editor of a special issue of IEEE Internet Computing on "Cloud Computing" (2009-2010), and translator for the greek language of the book "Google's PageRank and Beyond: The Science of Search Engine Rankings". His research interests are in the area of distributed systems, including the Web and Internet (conventional and streaming media distribution), social networks (ranking and mining), mobile and pervasive computing (indexing, caching, broadcasting) and wireless ad hoc and wireless sensor networks (clustering, routing, data dissemination, indexing, storage in flash devices).



George Pallis received his BSc and Ph.D. degree in Department of Informatics of Aristotle University of Thessaloniki (Greece). Currently, he is Marie Curie Fellow at the Computer Science Department, University of Cyprus. He is member of the High-Performance Computing Systems Laboratory. He is recently working on Web data management and he has focused on Web data caching, content distribution networks, information retrieval and Web data clustering. He has published several papers in international journals (e.g., CACM, IEEE Internet Computing, WWWJ etc.) and conferences (IDEAS, ADBIS, ISMIS etc.). He is co-editor of the book "Web Data Management Practices: Emerging Techniques and Technologies" published by Idea Group Publishing and co-guest editor of a special issue of IEEE Internet Computing on "Cloud Computing" (2009-2010).

Konstantinos Stamos was born in 1983 in Thessaloniki, Greece. In 2005 he graduated from Computer Science Department at Aristotle University of Thessaloniki, Greece. He received his Msc in 2007 and currently he is a PhD student. His main research interests are focused on Web data management, Web graph mining, Content distribution and delivery over the Web, Web data clustering, replication and caching.

Athena Vakali received her Ph.D. in Informatics from the Aristotle University, her M.S. in Computer Science from Purdue University and her BSc in Mathematics from the Aristotle University. She is currently an associate professor in the Department of Informatics at the Aristotle University, Thessaloniki, where she works as a faculty member since 1997. She is the head of the Operating Systems Web/INternet Data Storage and management research group. Her research activities are on various aspects and topics of the Web information systems, including Web data management (clustering techniques), content delivery on the Web, Web data clustering, Web caching, XML-based authorization models, text mining and multimedia data management. Her publication record is now at more than 100 research publications which have appeared in several journals (e.g CACM, IEEE Internet Computing, IEEE TKDE, WWWJ), book chapters and in scientific conferences (e.g. IDEAS, ADBIS, ISCIS, ISMIS etc). She is a member of the editorial board of the Computers and Electrical Engineering Journal (Elsevier), the International Journal of Grid and High Performance Computing (IGI), co-guest editor of a special issue of IEEE Internet Computing on "Cloud Computing" (2009-2010) and since March 2007, she is the coordinator of the IEEE TCSC technical area of Content Management and Delivery Networks. Prof. Vakali has leaded many research projects in the area of Web data management and Web Information Systems.

Antonis Sidiropoulos was born in Komotini, Greece in 1973. He received a BSc and a MSc in Computer Science from University of Crete (1996 and 1999, respectively), and a Ph.D. from Aristotle University of Thessaloniki, Greece (2006). Currently, he is a postdoc researcher at the Department of Informatics of Aristotle University of Thessaloniki. His research interests are in the areas of Web information retrieval and data mining, Webometrics, bibliometrics, scientometrics.

Yannis Manolopoulos was born in Thessaloniki, Greece in 1957. He received a B. Eng (1981) in Electrical Eng. and a PhD (1986) in Computer Eng., both from the Aristotle Univ. of Thessaloniki. Currently, he is Professor at the Department of Informatics of the latter university. He has been with the Department of Computer Science of the Univ. of Toronto, the Department of Computer Science of the Univ. of Maryland at College Park and the University of Cyprus. He has published over 160 papers in refereed scientific journals and conference proceedings. He is co-author of the books "Advanced Database Indexing", "Advanced Signature Indexing for Multimedia and Web Applications" by Kluwer and of the books "R-Trees: Theory and Applications", "Nearest Neighbor Search: A Database Perspective" by Springer. He is also author of two textbooks on Data Structures and File Structures, which are recommended in the vast majority of the computer science/engineering departments in Greece. His research interests include access methods and query processing for databases, data mining, and performance evaluation of storage subsystems.