On-Line Storage Management with Distributed Decision Making for Content-Centric Networks

Vasilis Sourlas*†, Lazaros Gkatzikis*† and Leandros Tassiulas*†
*Department of Computer & Communication Engineering,
University of Thessaly, Greece. †CERTH-ITI
Email: vsourlas, lagatzik, leandros @inf.uth.gr

***Abstract*—Content traffic proliferation in Internet makes more dire than ever the development of radical new network architectures, where information will be addressed by semantic attributes rather than the origin and destination identities. In this direction, content-centric networking appears as a flexible communication model that meets the requirements of the content distribution trends of the future Internet. In such networks, information will reside at various locations/nodes (the Content Delivery Network surrogate servers) and the requests of the users for some piece of information will be directed to the closest replica. Since the location of the users and the popularity of the content varies over time, the problem of finding the optimal replication pattern for the available content, given the storage constraints, comes into the foreground. In this paper, we propose two on-line storage management algorithms of gradient descent type, designed specifically for content-centric networks. The proposed algorithms are of polynomial complexity and thus adapt easily to any environmental changes. Each node re-assigns its information items with the aim to minimize the overall traffic cost of the content delivery as the popularity and locality of users' requests change. While both the proposed algorithms operate in a distributed way, differ in the amount of information required for the decision making. Thus, we identify the inherent *information–performance tradeoff* and compare them in terms of network traffic, convergence speed and amount of circulated information.**

I. Introduction

Distributed storages are placed at various points in a network so as to maximize bandwidth for access to the data from clients throughout the network. Each client accesses its closest replica of the data, as opposed to all clients accessing the origin server, in order to avoid bottleneck near that server.

Existing work on Content Delivery Networks (CDNs) and Web caches focuses on techniques for efficiently redirecting user requests to the appropriate replica to reduce request latency and balance traffic across nodes. Some attention has also been given to the development of placement strategies for the content available within a network to further improve CDN performance. However, the development of a storage management strategy that re-assigns the content among the CDN servers, as the popularity and the locality of the demand for the content change, has received only limited attention.

The content-centric publish/subscribe (pub/sub) paradigm is considered a promising future network architecture that solves many challenges of the current Internet and is appropriate for designing distributed systems due to its loose-coupled and asynchronous communication. In this paper, we present two on-line storage management algorithms for a network that operates under the content-centric pub/sub paradigm.

Our algorithms, given an initial storage configuration, re-assign the content among the storages with the aim to minimize the total traffic cost in the network. Due to storage limitations it is obvious that in order to store an item in a storage we may need to remove another one. Whereas, removing an item increases the total traffic cost, inserting a new one decreases it. The whole procedure is further complicated by storing correlations, meaning that storing of an item at one storage in the network affects the performance gain in traffic cost of storing the same item in other storages. The objective is to minimize the total traffic cost/load of all items in the network given that storages have storing limitations and items are of different size.

We also consider the special case where each storage node mainly focuses on the traffic of its clients (clients attached to it directly or belonging to the subnetwork it is responsible for) and attempts to minimize their traffic cost. Such a scenario may arise in the context of inter-connected CDNs of different providers. In this case each provider is primarily concerned about his own clients and offers his services to any others in a best-effort manner.

The rest of the paper is organized as follows. Section II briefly surveys previous work, while in Section III we formulate the storage management problem. In Section IV we present the two distributed on-line storage management algorithms. Moreover, in Section V, we evaluate through simulations the performance of the proposed algorithms. Finally, in Section VI we conclude our paper and give pointers to future work.

II. Related work

The storage placement problem, especially in the area of CDNs and Web caching, is a well investigated problem. Actually the placement problem is in fact an NP-hard problem, but there is a number of studies [1]-[5] where an approximate solution is pursued. Their work is also known as network location or partitioning and involves the optimal placement of k service facilities in a network of V nodes targeting the minimization of a given objective. This work is also known as the *k-median* problem. More replication placement strategies have been proposed in [6]-[7], where their main outcome is that the best results are obtained with algorithms

that have all the storages cooperating to make the replication decision. Finally, in [8]-[9] authors formulate the problem as a mixed integer program (MIP) and in order to overcome the challenges of scale they employ a Lagrangian relaxation-based decomposition technique combined with integer rounding.

In the area of Web caching, the Internet Cache Protocol (ICP) [10] and the HTCP [11] protocol support discovery, retrieval and management of documents from neighboring caches as well as parent caches. While most of the above approaches have a similar cost function (optimize bandwidth and/or storage usage costs for a given request pattern), less attention has been given to network constraints (limited storage capacity) and the possibility of re-assigning items between storages when the popularity and the locality of users' requests change.

There are several research efforts that develop an overlay event notification service like Siena [12], Elvin [13] and REDS [14] that implement the pub/sub architecture. Moreover, there are also several research efforts aiming to switch from host-oriented to content oriented networking to meet data-intensive application needs. In particular, NDN [15]-[16], PURSUIT [17] and SAIL [18] attempt to name data/content instead of naming hosts in a way to achieve scalability, security and performance. Finally, the pub/sub architecture paradigm is already used in many commercial applications such as Google Alerts through the usage of a simple, open, server-to-server web-hook-based pub/sub protocol called PubSubHubbub [19].

In the area of storing in overlay content-centric pub/sub systems the authors of [20] proposed a historic data retrieval pub/sub system where databases are connected to various nodes, each associated with a topic to store. There, every message is stored only once and no placement strategies have been examined. Finally, a first attempt with an off-line replication algorithm in topic-based pub/sub networks is presented in [21].

III. Problem formulation

Consider a network of arbitrary topology represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of storage nodes and $\mathcal{E}$ the set of directed communication links interconnecting these storages. Throughout the paper we will use the calligraphic letters to denote sets and the corresponding capitals for cardinality; for example $|\mathcal{V}| = V$.

Information is residing at different storages and requests for content access are generated at various nodes. We assume a set $\mathcal{M}$ of M information items and we denote with s^m the size of item m. Each storage $v \in \mathcal{V}$ has a storage capacity of C_v. We constrain ourselves to feasible network instances, i.e. we assume that at least one storage has sufficient storage capacity to fit any item ($C_v \geq s^m \quad \forall m \in M, v \in V$). Access requests trigger the transfer of the requested item from a node where the item is stored to the node where the request was generated. The performance of such a scheme is affected by the mechanism of deciding which replica of the requested item to transfer, as well as by the information replication scheme, i.e. how many replicas of each information item we maintain and where they are located. We consider the problem of optimizing the performance of such a scheme over the ensemble of possible storage configurations. The performance metric under consideration is the total traffic generated by the information access process. Specifically, the access of an information item m stored in node v by node y generates a traffic load equal to the product of the length (number of hops) of the path from y to v (d_{yv}) and the size s^m of the transferred item. In this paper we assume that an underlying content delivery mechanism optimally directs the requests to the closest storage out of those holding the specific item. Given such an access mechanism in order to minimize the total network traffic, we need to find the replication frequency/density and the location where each item should be stored.

We denote by $\mathcal{H}$ the collection of all possible storage configurations. That is, an element $\boldsymbol{H} \in \mathcal{H}$ can be represented by a binary matrix of size $V \times M$, specifying thus the content of each storage in the network. Actually, any element H_{vm} can be thought of as an indicator whether content item m exists in the storage v. Let $T(\boldsymbol{H})$ be the traffic load corresponding to configuration $\boldsymbol{H}$. We also denote by $\boldsymbol{h}^m$, the storage configuration regarding item m, namely the m-th column of matrix $\boldsymbol{H}$. Throughout this paper we denote with $f^m = \sum_{i=1}^{V} H_{im}$ the replication frequency of item m.

For a storage configuration $\boldsymbol{H}$ we can write:

$$T(\boldsymbol{H}) = \sum_{m=1}^{M} T^m(\boldsymbol{h}^m)$$

where $T^m(\boldsymbol{h}^m)$ is the traffic load corresponding to the configuration of item m only. We should mention here that throughout this text in the function definitions we include only the control variables as arguments and omit any network parameters.

We also have

$$T^m(\boldsymbol{h}^m) = s^m \sum_{v \in \mathcal{V}: H_{vm}=1} \sum_{y \in \mathcal{N}_v^m} r_y^m d_{yv} \tag{1}$$

where $\mathcal{N}_v^m$ is the set of nodes accessing item m through its replica at storage node v, r_y^m the request rate for information item m generated at node y, d_{yv} is the distance (in hops) from node y to node v and s^m the size of item m. Thus, the overall network traffic can be expressed as

$$T(\boldsymbol{H}) = \sum_{m=1}^{M} s^m \sum_{v \in \mathcal{V}: H_{vm}=1} \sum_{y \in \mathcal{N}_v^m} r_y^m d_{yv} \tag{2}$$

So, the problem of finding the optimal storage configuration can be formally expressed as the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{H}}{\text{minimize}} \quad & T(\boldsymbol{H}) \\ \text{s.t.} \quad & f^m \geq 1 \qquad \forall m \in M \\ & \sum_{m=1}^{M} s^m H_{vm} \leq C_v \quad \forall v \in V \end{aligned} \tag{3}$$

The set of constraints described above defines the feasible storage configurations. In particular, the first one indicates that each content item has to be stored at least once. Otherwise the total traffic would become unbounded. The second one captures the requirement that each storage needs to have a storage capacity such that it can store any message. The third one describes the fact that the messages that can be stored in each storage are limited by its storage capacity.

Actually, the optimal assignment of the items in the storages can be mapped to the Generalized Assignment Problem, which even in its simplest form may be reduced to an NP-complete multiple knapsack problem [6]. In this work we propose two algorithms of polynomial complexity that capture the particularities of content centric networks and can be executed in a distributed manner.

IV. Distributed decision making for on-line storage management algorithms

Our on-line storage management algorithms capture the particularities of the volatile environment under consideration and quickly adapt to any popularity and locality changes of the users' requests. All the required decisions are made in a distributed manner by each node. The key distinguishing characteristic between the two proposed algorithms is their objective and the amount of information required for the decision making. The first one, henceforth called *holistic*, aims at minimizing the traffic load of the network in total. According to the second one, hereafter called *myopic*, each storage node considers only the traffic cost incurred to its own clients. A detailed description of the proposed algorithms follows.

A. Holistic storage management algorithm

Given an initial $\boldsymbol{H} \in \mathcal{H}$ feasible storage configuration, the nodes of the network, independently and asynchronously, update the contents of their storages towards the global objective. The holistic approach can be thought of as a sequence of Gauss-Seidel iterations [27], where in each iteration a node, say $v \in \mathcal{V}$, executes the following steps:

Step 1: For each item m, stored in storage v and at least once more in the network (i.e. $f^m \geq 2$), compute the overall performance loss, $l_m = |T(\boldsymbol{H}) - T(\overline{\boldsymbol{H}}_m)| \geq 0$, that will be caused if item m is removed from v, leading hence to a new configuration $\overline{\boldsymbol{H}}_m$. In this case all the requests for item m at storage v will be served by another storage, which is at least that far.

Step 2: For each item m not in storage v, compute the overall performance gain $g_m = T(\boldsymbol{H}) - T(\underline{\boldsymbol{H}}_m) \geq 0$ achieved if item m is inserted at storage v, leading hence to a new configuration $\underline{\boldsymbol{H}}_m$. In this case a certain amount of requests for item m will be served by node v, as the closest replica.

Step 3: Consider as candidate for insertion, the item of maximum performance gain; say $g^* = \max(\boldsymbol{g}) = g_a$.

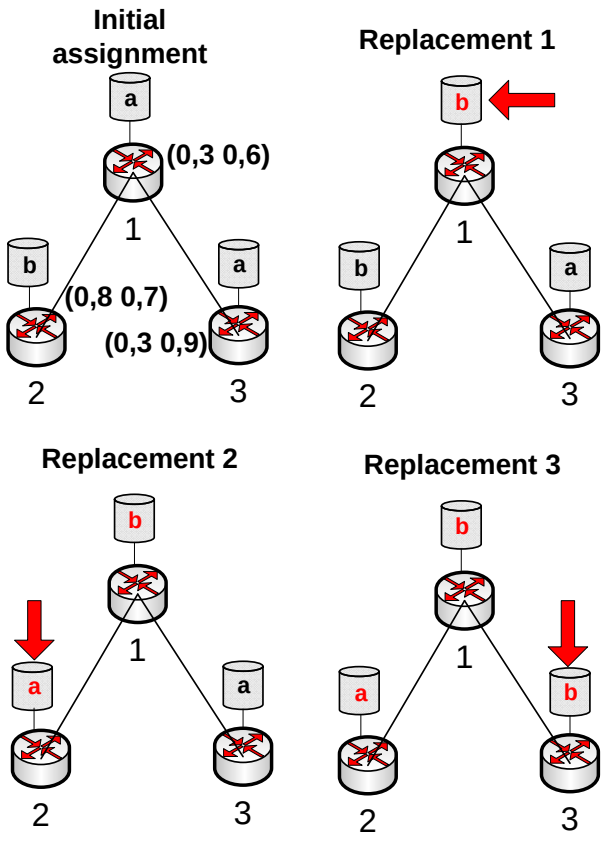


Fig. 1. Evolution of the storage management algorithm for a network of $V = 3$ nodes, $C_v = C = 1$ slots/node and $M = 2$ items with the depicted clients' interest rates.

Step 4: Consider as candidates for replacement the items of minimum performance loss. Starting from the minimum one (say $l^* = \min(\boldsymbol{l}) = l_b$), and in ascending order consider that many items that their total size is greater or equal to s^a (say b and c cause the minimum increase of the overall traffic with $s^a \leq s^b + s^c$).

Step 5: If the gain of storing the new item is greater than the loss of removing the selected items, perform the replacement, (i.e. *replace* b, c with a). In order to store this particular item (a), the node needs to retrieve it. Thus, it sends a subscription message to the closest replica.

Step 6: If the replacement leaves some free space (e_v) in the storage v ($e_v = C_v - \sum_{m=1}^{M} s^m H_{vm} > 0$) compute, for each item m not in storage v of size $s^m \leq e_v$, the performance gain g_m. Store the fitting item of maximum gain. Repeat until no other items could be stored, due to insufficient free space.

Step 7: Repeat steps 1-6 until no further replacements are beneficial for the network.

Figure 1 depicts an example of the algorithm's evolution for a network topology of $V = 3$ nodes, $C_v = C = 1$ slot and

$M = 2$ items of equal size. In brackets are the request rates for items a and b. The algorithm requires two iterations per node to reach an equilibrium. The top left side of the figure depicts the initial (random) storage configuration, whereas the bottom right side is the final storage configuration. In this case the resulting equilibrium coincides with the optimal assignment. Next, we describe the evolution of the algorithm for this simple network in detail, where T is the overall network traffic in $reqs \cdot hops/sec$:

- **node 1:** $T = 6.8$, $l_a = 1.1$, $g_b = 1.5$.
  Replace item a with item b, new $T = 6.4$.
- **node 2:** $T = 6.4$, $l_b = 0.7$, $g_a = 1.6$.
  Replace item b with item a, new $T = 5.5$.
- **node 3:** $T = 5.5$, $l_a = 0.6$, $g_b = 0.9$.
  Replace item a with item b, new $T = 5.2$.
- **node 1:** $T = 5.2$, $l_b = 1.3$, $g_a = 0.6$.
  No Replacements.
- **node 2:** $T = 5.2$, $l_a = Inf$ (only replica of item a), $g_b = 0.7$. No Replacements.
- **node 3:** $T = 5.2$, $l_b = 0.9$, $g_a = 0.6$.
  No Replacements. Algorithm terminates.

The essence behind this algorithm is that each node performs only valid and beneficial replacements, i.e replacements that lead to feasible storage configurations and improve the overall objective respectively. We say that the algorithm has reached an equilibrium point when no more replacements are possible.

It can be easily shown that since all the changes in the configuration of the storages always decrease the overall network cost, the proposed algorithm finally converges to an equilibrium point where no further decrease is possible. This is indeed a local minimum of the cost function but the performance gap to the global optimum cannot be calculated. Besides, though the storage updates may be applied asynchronously among the nodes, we assume that only a single node may modify the storage configuration at a given time. This is because each node needs to know the current storage configuration of the network as a whole, so as to calculate the gain and loss metrics. After each modification the node advertises the new configuration of its own storage. Relaxing this assumption would lead to a setting where the nodes make decisions based on outdated information, causing thus some performance degradation and making convergence questionable.

In appendix A we present in pseudo-code the Holistic storage management algorithm.

B. Myopic storage management algorithm

In the holistic approach described earlier every node needs to possess global information for the decision making. However, in highly dynamic environments the amount of information that needs to be circulated throughout the network grows, causing thus significant communication overhead. Even worse the required information may not be available on time, making hence such an approach inapplicable.

For such scenarios we derive an alternative approach. We assume that each node has no information about the rate patterns at the other storages, and thus makes decisions based only on local information. That is each node v has to select his own storage configuration $\boldsymbol{H}_{vm}$ for $m = 1 \dots M$, so as to minimize the traffic cost of his own clients. Thus, its objective function now becomes the following:

$$T_v(\boldsymbol{H}) = \sum_{m=1}^{M} s^m r_v^m d_{vy_m}, \tag{4}$$

where y_m the closest replica of item m. Given the new objective function, though the performance gain and loss expressions change, the main steps of myopic algorithm remains the same with the holistic one. For brevity, we do not present the myopic algorithm in detail; we highlight their differences in the section that follows.

The myopic algorithm captures also the scenario of interconnected CDNs of different providers, where each provider is primarily concerned with his own clients and offers his services to the others in a best-effort manner.

C. Information exchange in the pub/sub paradigm

In this section we focus on the information exchange required for the proper execution of our algorithms and highlight their distinguishing characteristics. The main difference between the two approaches lies in the amount of information exchanged. In particular, since the myopic one requires less information for the decision making, it is applicable to more network scenarios and incurs less communication overhead.

Both approaches require the current storage configuration to be known at every node. Actually, every node needs to know the size of each item and the cost to the closest replica. Such information is available at contemporary CDNs [6] and could be easily provided by a pub/sub architecture. Moreover, Web caches can keep a digest [22] or summary [23] of the content of the other caches in the network.

In addition, the holistic approach requires the request rates per item to be globally known. The request rates can be estimated based on recent request history, which is locally available [24] and nodes periodically exchange metadata information regarding the request patterns. This information can be forwarded to every node of the network using a special purpose *publish* packet, flooded throughout the network. The size of such a packet is typically a few tens of bytes and is negligible compared to the size of an item. For environments where the request rate pattern varies insignificantly or slowly over time, the required frequency of flooding will be quite small. Hence, the memory and the network traffic overhead of maintaining and exchanging those packets will be quite small.

In the myopic approach, on the other hand, each node makes its decisions based only on local information, namely the request rates of its own clients. Thus, this algorithm fits best to highly volatile scenarios, where the timescale of storage updates is comparable to the timescale that the request rate patterns change. In such a setting the holistic approach would

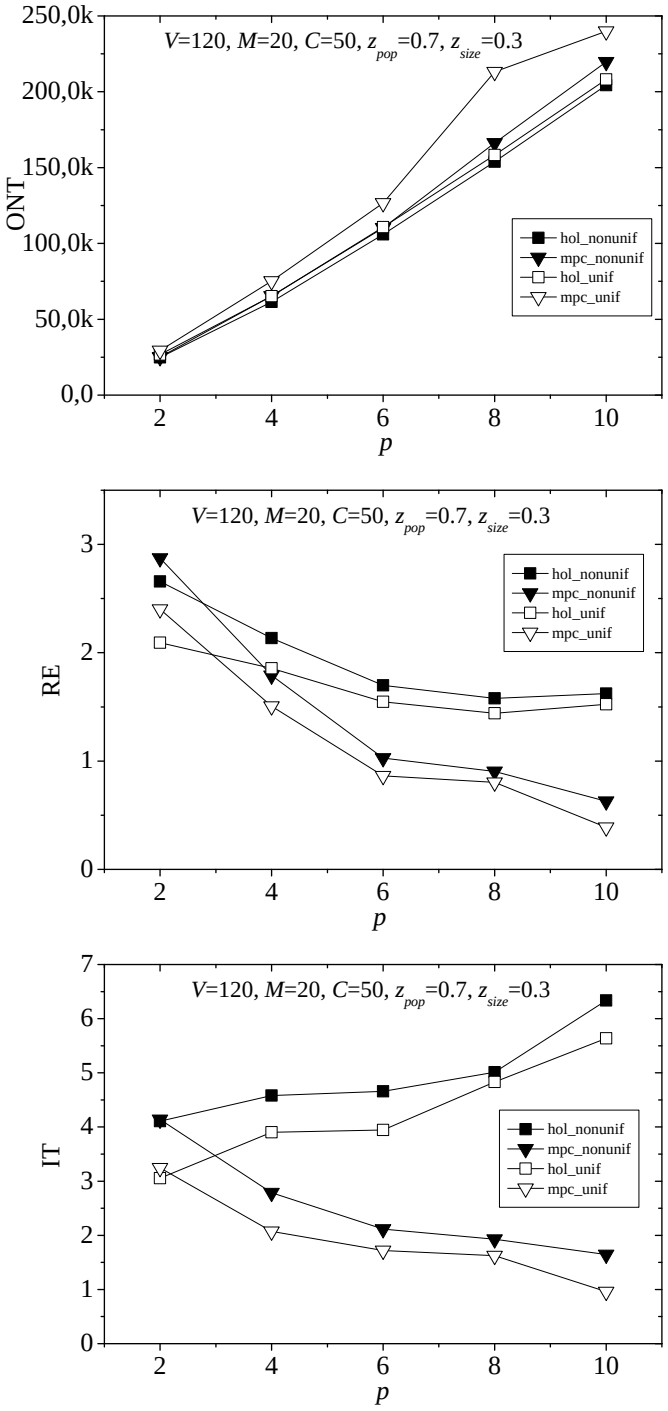


Fig. 2. The performance of the proposed storage management algorithms vs. the ratio of the total size of the items to storage capacity ($p = \sum_{m=1}^{M} s^m/C$) for our random topology of 120 nodes.

always operate in the transient phase and convergence would not be achieved. Besides, the incurred overhead cost would become significant.

V. Performance evaluation

In order to evaluate the performance of the proposed storage management algorithms we simulate in MATLAB a random network topology of $V = 120$ storage nodes, serving a set of one-hop clients each. Our graph also contains cycles. Each node has a storage capacity of $C = 50$ storage units and we assume a set of $M = 20$ information items. Throughout this section, we will use the abbreviations *hol* and *mpc* for the *holistic* and the *myopic* approach respectively.

Initially, we assume uniform request rates for each item at each storage node. That is, the traffic request pattern of the clients of storage node y for item m is $r_y^m = r^m$, the same $\forall y \in \mathcal{V}$. The request rates lie between 0 and 25 requests/sec and the popularities of the items come from a Zipf-like distribution (of default parameter $z_{pop} = 0.7$), whereas the size of each item also follows a Zipf distribution (of default parameter $z_{size} = 0.3$). Regarding popularity, values between 0.6 and 0.8 have been observed in the Web [25]-[26]. Next, we consider the impact of non-uniform request rates per item for each node of the network.

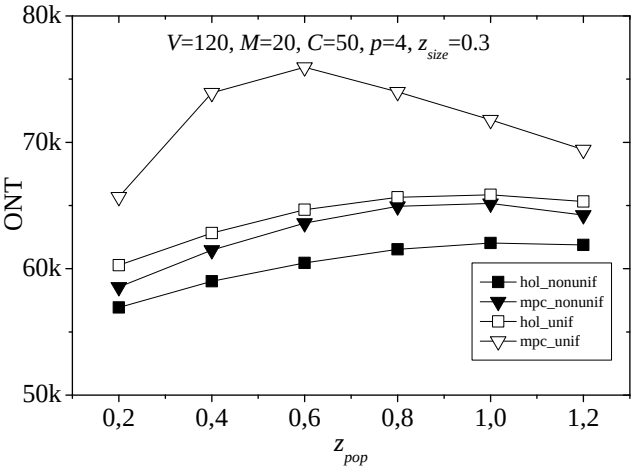
For both scenarios, we perform a set of experiments to study the impact of the following parameters on the performance of the proposed algorithms:

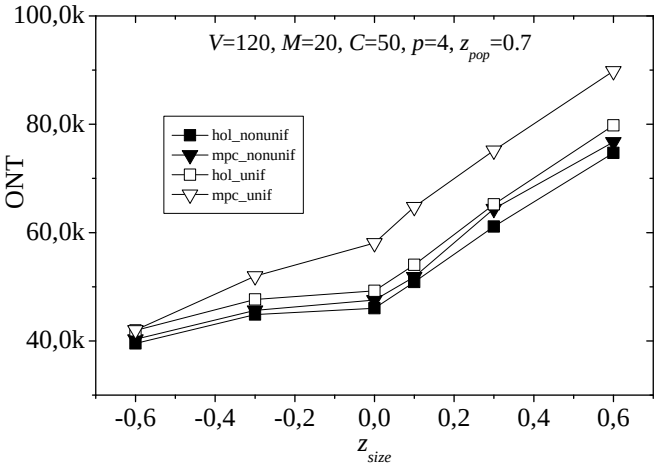
- the ratio of the total size of the items to storage capacity (p). Under the assumption of equal capacity storages, i.e. $C_v = C \ \forall v \in \mathcal{V}$, this can be expressed as $p = \sum_{m=1}^{M} s^m/C$. From an alternative point of view $1/p$ is the fraction of items that can be stored in a cache.
- The exponent of the popularity of the items z_{pop}.
- The exponent of the size of the items z_{size}.

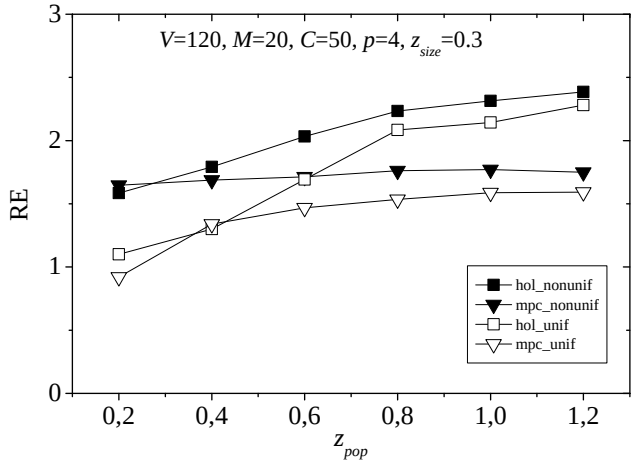
In order to quantify the performance of the proposed algorithms we consider the following metrics:

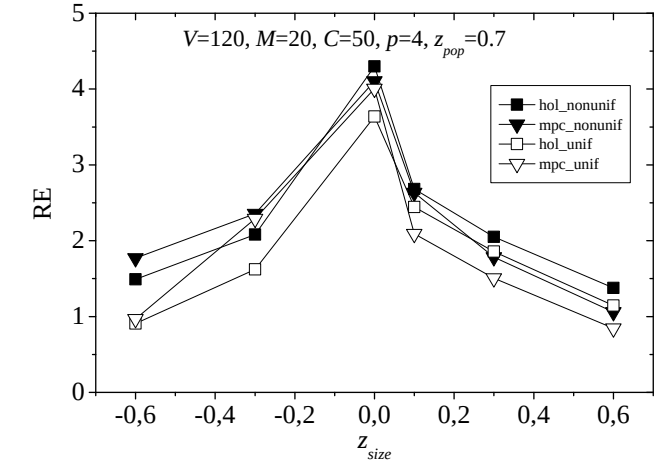
- The *overall network traffic* (ONT) (in $req \cdot hops/sec$) at the equilibrium.
- The *total number of replacements per storage* (RE) required to reach the equilibrium.
- The *total number of iterations per node* (IT) required to reach the equilibrium. IT is the number of times that each storage has to execute the proposed algorithms, performing beneficial replacements until an equilibrium is reached.

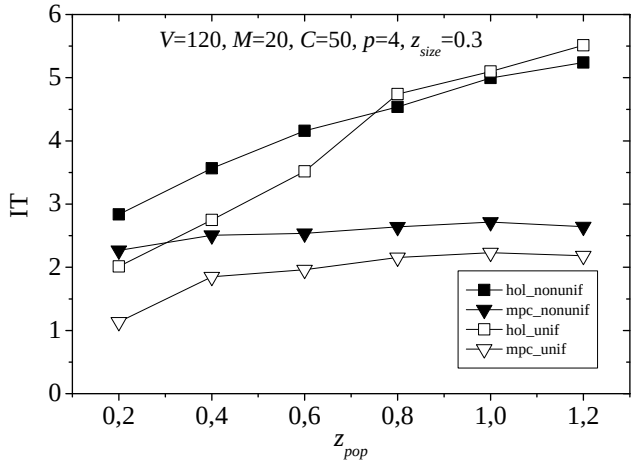
Each point of the figures depicts the average out of 20 random instances, starting from different initial storage configurations $\boldsymbol{H}$ for the particular topology defined earlier. Figure 2 depicts the performance of the proposed storage management algorithms as an expression of the percentage metric p. As expected, the holistic approach outperforms its myopic counterpart by $1\% - 8\%$ regarding the ONT when the request rates are non-uniformly distributed to the storage nodes, and by $9\% - 34\%$ when the request rates are uniformly distributed. This is due the additional (global) information that is available in the holistic approach. Both the absolute performance and the performance gap become more significant as percentage of stored messages decreases. In this point, we should mention that the uniform and non-uniform plots are











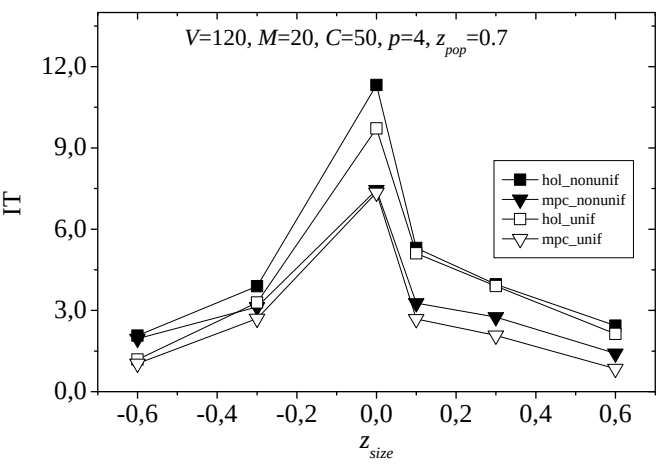


Fig. 3. The performance of the proposed storage management algorithms vs. the popularity exponent of the items z_{pop} for our random topology of 120 nodes.

Fig. 4. The performance of the proposed storage management algorithms vs. the popularity exponent of the items z_{size} for our random topology of 120 nodes.

not directly comparable, since they actually refer to different network instances, but are included in the same graph for brevity.

Regarding the average number of replacements the holistic approach requires on average $10\% - 150\%$ more replacements from each storage node to reach the equilibrium point. A replacement in the holistic algorithm usually affects most of the nodes in the network triggering thus a new round of replacements. On the other hand a replacement in the myopic algorithm affects mainly the node where the replacement took place and probably a small number of neighboring nodes. Similar behavior we may notice regarding the average number

of iterations. Besides, in the holistic algorithm the IT metric increases along with p, in contrast to the myopic approach, where a decrease is noticed. This is also a consequence of the topological impact that a replacement has to the two storage management algorithms. In particular the myopic approach by considering only local information is less sensitive to changes of the environment. This becomes more obvious as the storage capacity becomes limited, where, given the local decision making, beneficial replacements are scarce. The product of the IT and RE metrics is also indicative of the computational complexity of the two algorithms.

Figure 3 depicts the performance of the proposed storage management algorithms as the popularity exponent of the items varies. The holistic algorithm outperforms the myopic one by $2\%-5\%$ in the non-uniform scenario and by $6\%-18\%$ in the uniform scenario. We notice that the performance gap is quite insensitive to any popularity changes. Regarding the complexity, both the RE and IT metrics increase along with the popularity exponent, but the myopic algorithm is quite robust to any popularity changes, leading thus much faster to the equilibrium.

Finally, in Figure 4 we consider the impact of the size exponent. Negative values of the z_{size} mean that the less popular items are those that have larger size. As the size of popular files increases, the performance of the system degrades since more storage space is required to store a popular item. However, the myopic approach manages to perform quite well in any scenario, especially for the non-uniform case. The IT and RE metrics indicate a peak for z_{size} equal to zero (i.e. the items are of the same size). This can be justified by the fact that equally sized files make decision making more challenging, requiring thus more iterations to reach an equilibrium. In particular, in this case the total number of iterations and replacements is at least double, since more items are candidates for replacement and each change leads to a small improvement.

VI. Conclusion And Future Work

In this paper, we presented two distributed, on-line gradient descent type storage management algorithms, designed specifically for content-centric networks, that re-assign stored items with the aim to minimize the total traffic cost. Since the proposed algorithms are of gradient descent type, they always converge to an equilibrium, where no further reduction in the overall network traffic is possible. However, fast convergence is required in order to adapt to any changes regarding the popularity and the locality of requests. In particular, the first one (holistic) tries to minimize the total traffic cost/load of all items in the network given that storages have storage limitations and items are of different size, while in the second one (myopic) each node tries to minimize the traffic of its own clients only. We also derive structural properties of the optimal assignment and evaluate the performance of the proposed storage management algorithms through simulations.

Our simulations indicate that the holistic algorithm performs better than the myopic regarding the overall network traffic at the equilibrium, regardless of the storage capacity of each storage and the Zipf's-law exponent of the items' size and popularity. However, this comes at the cost of increased complexity and slower convergence rate, since in general it requires more replacements of items and iterations of the algorithm to reach the equilibrium. It would be interesting, as future work, to explore enhancements to the proposed algorithms that would also take into consideration the cost of replacing the items at the storages of the network, as well as the load of each storage when assigning items to them. Finally, a comparison to the optimal performance for at least some small network instances will be pursued.

Acknowledgment

V. Sourlas' and L. Gkatzikis' work was supported by the Greek Ministry of National Education and Religious Affairs (E.S.P.A.- "HRAKLEITOS II"). L. Tassiulas' contribution is funded by the European Commission through the FP7-ICT-2010-257217 PURSUIT and the FP7-ICT-216366 EURO-NF programs.

References

[1] L. Qiu, V.N. Padmanabhan, G. Voelker, "On the placement of web server replicas," In IEEE INFOCOM, Anchorage, USA, Apr. 2001.
[2] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala and V. Pandit, "Local search heuristics for k-median and facility location problems," In 33rd ACM Symp. on Theory of Computing, 2001.
[3] M. Charikar, S. Guha, "Improved combinatorial algorithms for facility location and k-median problems," In 40th Annual IEEE Symp. on Foundations of Computer Science, pp. 378–388, Oct. 1999.
[4] D.B. Shmoys, E. Tardos, K.I. Aardal, "Approximation algorithms for facility location problems," In 29th Annual ACM Symp. on Theory of Computing, pp. 265-274, 1997.
[5] E. Cronin, S. Jamin, C. Jin, T. Kurc, D. Raz, Y. Shavitt, "Constrained mirror placement on the Internet," IEEE JSAC, 36(2), Sept. 2002.
[6] J. Kangasharju, J. Roberts, K. Ross, "Object replication strategies in content distribution networks," Comput. Commun. 25 (4) (2002) pp. 376-383.
[7] X. Tang, J. Xu, "On replica placement for QoS-aware content distribution," In IEEE INFOCOM, March 2004.
[8] I.D. Baev, R. Rajaraman, C. Swamy, "Approximation algorithms for data placement problems," SIAM J. Computing, vol. 38, 2008.
[9] D. Applegate, A. Archer, V. Gopalakrishnan, "Optimal content placement for a large-scale VoD system," In ACM CoNEXT, Philadelphia, USA, Dec. 2010.
[10] D. Wessels, K. Claffy, "Applications of Internet Cache Protocol (ICP), v.2," http://tools.ietf.org/html/draft-wessels-icp-v2-02, ITF, May 1997.
[11] P. Vixie, D. Wessels, "RFC 2756: Hyper Text Caching Protocol," Jan. 2000.
[12] A. Carzaniga, D. Rosenblum, A. Wolf, "Design and evaluation of a wide-area event notification service," ACM Trans. On Computer Systems, vol. 19, 2001.
[13] B. Segall, D. Arnold, "Elvin has left the building: A publish/subscribe notification service with quenching," In AUUG97, Brisbane, Australia, Sept. 3-5, pp. 243–255, 1997.
[14] G. Cugola, G. Picco, "REDS, A Reconfigurable Dispatching System,"' In 6th Inter. workshop on Software Engineering and Middleware, pp. 9–16, Oregon, 2006.
[15] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, R. Braynard, "Networking named content," In. ACM CoNEXT, Rome, Italy, Dec. 2009.
[16] Named Data Networking project, available at http://named-data.net, 2011.
[17] PURSUIT project, available at http://www.fp7-pursuit.eu, 2011.
[18] B. Ohlman et al, "First NetInf architecture description," April 2009. http://www.4ward-project.eu/index.php?s=file_download&id=39.
[19] PUBSUBHUBBUB, available at http://pubsubhubbub.googlecode.com

[20] G. Li, A Cheung, S. Hou, S. Hu, v. Muthusamy, R. Sherafat, A. Wun, H. Jacobsen, S. Manovski, "Historic data access in publish/subscribe," In DEBS, pp. 80–84, Toronto, Canada, 2007.
[21] V. Sourlas, P. Flegkas, G. S. Paschos, D. Katsaros, L. Tassiulas, "Storing and Replication in Topic-Based Publish/Subscribe Networks," In IEEE Globecom, Miami, USA, Dec. 2010.
[22] A. Rouskov, D. Wessels, "Cache Digest," In 3rd Inter. WWW caching workshop, June 1998.
[23] L. Fan, P. Cao, J. Almeida, A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," In SIGCOMM, pp. 254–265, Feb. 1998.
[24] J. Shim, P. Scheuermann, R. Vingralek, "Proxy Cache Algorithms: Design, Implementation, and Performance," IEEE Trans. Knowledge and Data Eng., vol. 11, no. 4, pp. 549–562, Aug. 1999.
[25] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," In IEEE INFOCOM, NY, March 1999.
[26] V. N. Padmanabhan, L. Qiu, " The content and access dynamics of a busy wed site," In SIGCOMM, Stockholm, Sweden, Aug. 2000.
[27] D. P. Palomar, M. Chiang, " A tutorial on decomposition methods for network utility maximization," In IEEE Journal on Selected Areas in Communications, 24(8), 1439-1451, 2006.

APPENDIX A
HOLISTIC STORAGE MANAGEMENT ALGORITHM

Require: $\boldsymbol{H}$: storage configuration
Require: f^m replication frequency of item m.
Require: M: number of the different items in the network
Require: V: number of nodes in the network topology
Require: C_v: storage capacity of node $v \in V$
$u \leftarrow V$
while $u > 0$ **do**
for all $v \in V$ **do**
for all $m \in M$ **do**
if $v \in \boldsymbol{h}^m$ **and** $f^m > 1$ **then**
CALCULATE l_m
INSERT l_m IN L_v
else $\{v \notin \boldsymbol{h}^m\}$
CALCULATE g_m
end if
end for
$I = 0$
$S = 0$
s^y {size of item y with the maximum gain (g_y)}
for all $j \in L_v$ **do**
if $S \geq s^y$ **then**
$I = I + min(L_v)$
$S = S + s^x$ {x item in L_v with me minimum loss $(min(L_v) = l_x)$}
INSERT x IN R {R items candidate to be replaced}
$l_x = Inf$ {$l_x \in L_v$}
end if
end for
if $I < g_y$ **then**
for all $k \in R$ **do**
DISCARD $R(k)$
end for
STORE y
$e_v = C_v - s^y + S$ {free space}
if $e_v > 0$ **then**
for all $m \in M$ **do**
if $H_{vm} = 0$ **and** $s^m \leq e_v$ **then**
CALCULATE g_m
INSERT g_m IN G
end if
end for
while $e_v > 0$ **do**
if $s^y \geq e_v$ **then**
STORE y {y the item in D with the maximum gain}
$d_y = 0$ {$d_y \in G$}
$e_v = e_v - s^y$
end if
end while
end if
UPDATE $\boldsymbol{H}$
$u \leftarrow V$
else
$u \leftarrow u - 1$
end if
end for
end while