# Autonomic Cache Management in Information-Centric Networks

Vasilis Sourlas*, Paris Flegkas*†, Lazaros Gkatzikis* and Leandros Tassiulas*†

* Department of Computer & Communication Engineering. University of Thessaly, Greece.
† Centre for Research and Technology Hellas - ITI.
Email: {vsourlas, pflegkas, lagatzik, leandros}@inf.uth.gr

***Abstract*—Recent research efforts in the area of future networks indicate Information-Centric Networking (ICN) as the dominant architecture for the Future Internet. The main promise of ICN is that of shifting the communication paradigm of the internetworking layer from machine endpoints to information access and delivery. Optimized content dissemination and efficient caching of information is key to delivering on this promise. Moreover, current trends in management of future networks adopt a more distributed autonomic management architecture where management intelligence is placed inside the network with respect to traditional off-line external management systems. In this paper, we present an autonomic cache management approach for ICNs, where distributed managers residing in cache-enabled nodes decide on which items to cache. We propose three online cache management algorithms with different level of autonomicity and compare them with respect to performance, complexity, execution time and message exchange overhead. Our extensive simulation-based experimentation signifies the importance of network wide knowledge and cooperation.**

## I. Introduction

ICN is emerging as the main future networking environment, given that the vast majority of Internet activities are related to information access and delivery. Thus, several research groups proposed and developed direct information-centric routing architectures, using location independent content ID/names instead of endpoint-node addresses; see NDN [1], PURSUIT [2] and SAIL [3]. In ICNs, information is explicitly labeled so that anybody, who possesses relevant information, can potentially participate in the fulfillment of requests for said information, making efficient caching of information items one of the most significant management tasks of future networks.

Moreover, the emerging requirements of the future networks coming from the proliferation of services deployed over the Internet such as interactive applications, telecommunication services, safety and mission critical systems and also its use for business and social interactions, all demand better quality, dependability, resilience and protection. Current management approaches based on off-line external management systems are inadequate towards these demands. As a result, there is a need for introducing self-management intelligence within the network in order to make the latter more flexible and adaptive to changing conditions through feedback closed-loop control solutions.

978-1-4673-0269-2/12/$31.00 

Autonomic self-management is intrinsic in IBM's pioneering autonomic computing vision [4] which envisages systems that can manage themselves given high-level objectives by administrators. Extending autonomic management from individual devices to the collective self-management of networks of such devices results in autonomic networking and there have been significant efforts in this area over the last five years. While a number of approaches have been proposed related to autonomic network management, most of them are generic high-level architectures [5]-[6] and design of infrastructures [7] focusing only partially on the key research issues. Most of the approaches are based on the concept of knowledge planes as introduced by [8] or intelligent substrates that realize a distributed infrastructure for embedding dedicated management logic in the network in a controllable manner.

Caching is common practice in current networking architectures as a means of improving performance as this is perceived from the users. Though finding the optimal cache management/placement is generally an NP-hard problem, several studies [9]-[11] propose approximate solutions. The authors of [12]-[13] formulate the problem as a mixed integer program (MIP) that takes into account constraints such as disk space, link bandwidth and content popularity. Moreover, [14] compares the performance of several caching architectures, and presents analytical models for important performance parameters of Web caching, such as clients perceived latency and bandwidth usage.

The authors of [15] model cache assignment as a distributed selfish replication (DSR) game in the context of distributed replication groups (DRG). Under the DRG abstraction, nodes utilize their caches to replicate information items and make them available to local and remote users. The pairwise distance of the nodes is assumed to be equal, while in our approach we consider the generic case of arbitrary distances. In the context of DRG and under the same distance assumption a 2-approximation cache management algorithm is presented in [16]. Finally in [17] authors develop a cache management algorithm aimed at maximizing the traffic volume served from the caches and minimizing the bandwidth cost. They focus on a cluster of distributed caches (Content Distribution Networks), either connected directly or via a parent node, and formulate the content placement problem as a linear program in order to benchmark the globally optimal performance.

While packet-level in-network opportunistic caching is one of the salient characteristics of ICN architectures, cache placement and replica assignment still has an important role to play. Particularly in [18] a set of off-line cache planning and replica assignment algorithms for the ICN paradigm is proposed. Unlike in-network opportunistic caching [19]-[20], which attempts to cache the most commonly accessed items/packets as close to the clients as possible, replication distributes a site's contents across multiple mirror servers. Replication is used to increase availability and fault tolerance, while it has as side effect load-balancing and enhanced client-server proximity [21]. In general, replication and in-network opportunistic caching complement each other. Essentially every major aspect of a caching scheme has its equivalent in replicated systems, but not vice versa [16].

In this paper, we propose an autonomic cache management architecture that dynamically (re-)assigns information items to caches in an ICN. Distributed managers make information item (re-)placement decisions, based on the observed item request patterns such as their popularity and locality, in order to minimize the overall network traffic imposed by the user requests. Moreover, we present three distributed on-line cache management algorithms, categorize them based on the level of cooperation needed between the autonomic managers and we compare them against their performance, complexity, message overhead and convergence time. We validate the proposed approaches through simulations providing also insight on their ability to adapt depending on the volatility of the user requests.

The rest of the paper is organized as follows. Section II presents the functionality of the management substrate that will coordinate the caches, while in Section III, we formulate the cache management problem. In Section IV, we present the three distributed on-line cache management algorithms, while in Section V the communication and computational complexity of the proposed algorithms is analyzed. Moreover, in Section VI, we evaluate through simulations the performance of the proposes algorithms and we compare their outcome with centralized off-line solutions and completely selfish techniques. Finally, in Section VII we conclude our paper and give pointers for future work.

## II. Autonomic Cache Management System Architecture

In the emerging ICN proposals, information is replicated almost ubiquitously throughout the network with subsequent optimal content delivery to the requesting users. Thus, efficient placement and replication of information items to caches installed in network nodes is key to delivering on this promise. When a client is interested in a particular piece of content, his/her request (interest in [19] or subscription in [2]) can be redirected to one of the existing replicas rather than requiring retrieval from the original publisher. Consequently, management of such networks entails managing the placement and assignment of information to caches available in the network with objectives such as minimizing the content access latency from clients, maximizing the traffic volume served by caches and thus minimizing bandwidth cost and network congestion.

Current approaches applied to Content Distribution Networks follow static off-line approaches with algorithms that decide the optimal location of caches and the assignment of information items and their replicas to those caches based on predictions of content requests by users. In contrast, we propose the deployment of an intelligent substrate architecture that will enable the assignment of information items to caches to take place in real-time, based on changing user demand patterns. Distributed *Cache Managers* decide the items every cache stores by forming a substrate that can be organized either in a hierarchical manner for scalability reasons or in a peer-to-peer organizational structure. Communication of information related to request rates, popularity/locality of information items and current cache configurations, takes place between the distributed cache managers through the intelligent substrate functionality.

Every cache manager, as depicted in Fig. 1, should decide in a coordinated manner with other managers whether to cache an item. This may require the replacement of an already stored item, depending on the available space at the cache. The decision of this replacement of stored items can be performed towards maximizing an overall network-wide utility function (e.g. the gain in network traffic) which means every node should calculate the gain the replacement of an item would incur. This approach assumes that every cache manager has a holistic network-wide view of all the cache configurations and relevant request patterns and this information should be exchanged periodically or in an event-based manner when a manager changes the configuration of its cache.

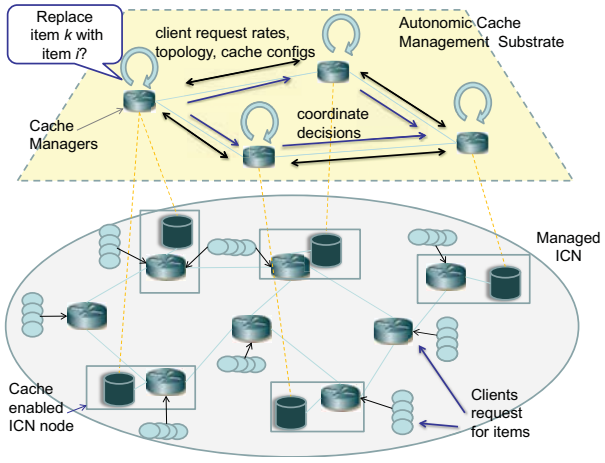


Fig. 1. Autonomic Cache Management Substrate in Information-Centric Networks

Other approaches can also be realized in which managers base their decisions on a local view of the user demand for specific items but coordinate to maximize the overall network gain, as well as solutions where managers act selfishly aiming at maximizing their own local utility. Since all the above decisions are made in a distributed manner, uncoordinated decisions could lead to suboptimal and inconsistent config-

urations. Coordinated decision making of a distributed cache management solution can be achieved through the substrate mechanisms, by ensuring that managers change the configuration of each cache in an iterative manner i.e. one at a time and not autonomously in a potentially conflicting manner. All these approaches are investigated in this work by proposing and comparing different distributed cache management algorithms and evaluating their performance with respect to their autonomicity. In the next sections, we present the cache management problem formulation followed by the description of the proposed algorithms.

## III. Cache Management Problem Formulation

We consider an information-centric network of arbitrary topology, represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ denotes the set of caches and $\mathcal{E}$ the communication links interconnecting them. Throughout the paper we will use the calligraphic letters to denote sets and the corresponding capitals for cardinality; for example $|\mathcal{V}| = V$.

We denote with $\mathcal{M}$ the set of the $M$ information items available at the network and with $s^m$ the size (in bits) of item $m$. The information items reside at the caches and requests for content access are generated by the users of the network, with each user being directly connected to a cache node, say its closest one. Cache $v \in \mathcal{V}$ has a storage capacity of $C_v$ bits and serves requests generated with rate $r_v = \{r_v^1, \ldots, r_v^M\}$, where $r_v^m$ denotes the aggregate incoming request rate (in requests per second) at cache $v$ for information item $m$. Access requests trigger the transfer of the requested item from a cache hosting the item to the node where the request was generated. A request by node $u$ for an item $m$ cached at node $v$ generates a traffic load equal to the product of the length $d_{vu}$ (in number of hops) of the path from $v$ to $u$ and the size $s^m$ of the transferred item.

We also denote by $\mathcal{H}$ the set of all possible cache configurations. A configuration $\boldsymbol{H} \in \mathcal{H}$ can be represented by a binary matrix of size $V \times M$ and specifies the content of each cache in the network. Actually, $H_v^m$ indicates whether information item $m$ is cached at $v$.

$$H_v^m = \begin{cases} 1 & \text{if item } m \text{ is cached at node } v, \\ 0 & \text{otherwise.} \end{cases}$$

In this paper, and in compliance with many ICN architectures, we assume that no origin server exists to serve requests for items not replicated at any cache. Thus, the following constraints define the set of feasible cache configurations:

$$\begin{aligned} \sum_{v=1}^{V} H_v^m &\geq 1 \qquad \forall m \in \mathcal{M} \\ \sum_{m=1}^{M} s^m H_v^m &\leq C_v \qquad \forall v \in \mathcal{V} \end{aligned} \tag{1}$$

In particular, the first one indicates that each content item has to be stored in at least one cache. Otherwise, the total traffic would become unbounded. The second one describes the fact that each cache has a limited capacity that cannot be exceeded.

An intelligent substrate architecture enables the assignment of information items to caches to take place in real-time, based on the ever-changing user demand patterns. In this study, we assume that the substrate is organized in a peer-to-peer fashion where distributed managers, one responsible for each cache, decide the items that should be stored in each cache according to specific performance criteria (see Fig. 1). Since there is a one-to-one mapping between managers and network caching nodes, we use the same notation $\mathcal{V}$ to denote both. Communication of information related to request rates, popularity/locality of information items and current cache configurations takes place between the distributed cache managers through the intelligent substrate functionality.

## IV. Distributed On-Line Cache Management Algorithms

In this section we present three distributed, gradient descent type, on-line cache management algorithms that capture the particularities of the volatile environment under consideration. All of them are adaptive to popularity and locality changes of the user demands. To achieve this each manager may update the contents of its corresponding cache, by fetching new items and replacing existing ones. We call this process item replacement. Nevertheless, the proposed mechanisms differ in the amount of information that needs to be communicated through the substrate, the required level of coordination among the cache managers, and the performance objective. We present them in order of decreasing complexity, in terms of the induced computational/communication overhead.

The first one, henceforth called *cooperative*, aims at minimizing the overall network traffic. This requires that every cache manager needs a holistic network-wide view of the request patterns and the current cache configuration. In addition, since each replacement decision affects the whole network, some cooperation in the decision making is required.

The second algorithm, henceforth called *holistic* also aims at minimizing the overall network traffic and hence requires the same amount of information. On the other hand, the holistic algorithm does not require coordination of the actions of the cache managers and the required decisions are made in an autonomous manner by each manager individually.

Finally, in the third algorithm, henceforth called *myopic*, each manager needs global knowledge of the items cached in the network, but only local knowledge regarding demand. Also, the myopic algorithm assumes that each manager acts autonomously and has the objective of minimizing the traffic related to its own demand.

Throughout the paper we assume that the underlying content delivery mechanism optimally directs the requests to the closest cache out of those holding the requested item. Given such an access mechanism in order to minimize the total network traffic, the cache managers have to coordinate their actions towards finding the replication frequency/density and the location where each item should be cached.

Without loss of generality, we also assume that each information item is of unit size ($s^m = s = 1$, $\forall m \in \mathcal{M}$) and all caches have the same storage capacity ($C_v = C$, $\forall v \in \mathcal{V}$). A detailed description of the proposed algorithms follows.

### A. Cooperative cache management algorithm

Let $T(\boldsymbol{H})$ be the traffic load corresponding to cache configuration $\boldsymbol{H}$.

$$T(\boldsymbol{H}) = \sum_{m=1}^{M} \sum_{\substack{v \in \mathcal{V}: \\ H_v^m = 1}} \sum_{u \in \mathcal{N}_v^m} r_u^m d_{vu} \tag{2}$$

where $\mathcal{N}_v^m$ is the set of nodes accessing item $m$ through its replica at cache node $v$, $r_u^m$ the request rate for information item $m$ generated at node $u$ and $d_{vu}$ is the distance (in hops) from node $v$ to node $u$.

Our objective here is to minimize the total traffic load in the network under the constraints of Eq. (1). However, finding the optimal assignment of the items in the caches, even for a static environment, can be mapped to the Generalized Assignment Problem, which in its simplest form is equivalent to the NP-complete multiple knapsack problem [22]. Thus, we propose the following adaptive heuristic mechanism, where at each iteration all the cache managers $v \in \mathcal{V}$ execute the following steps in parallel:

**Step 1:** For each item $m$ cached in $v$ compute the overall performance loss, $l_m = T(\overline{\boldsymbol{H}}_m) - T(\boldsymbol{H}) \geq 0$, that will be caused if item $m$ is removed from $v$, if this leads to a valid new configuration $\overline{\boldsymbol{H}}_m$. In this case all the requests for item $m$ at $v$ will be served by another cache, which is at least that far.

**Step 2:** For each item $m$ not cached in $v$ compute the overall performance gain $g_m = T(\boldsymbol{H}) - T(\underline{\boldsymbol{H}}_m) \geq 0$ achieved if item $m$ is inserted at cache $v$, 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:** Each manager $v$ considers as candidate for insertion item $i \in \mathcal{M}$ of maximum performance gain, i.e $i = \arg\max \boldsymbol{g}$ and as candidate for replacement the item $k \in \mathcal{M}$ of minimum performance loss i.e. $k = \arg\min \boldsymbol{l}$.

**Step 4:** Each manager $v$ calculates the local maximum relative gain $r = g_i - l_k$ and informs the rest of the cache managers through a report message $Rep(r, v, i, k)$.

**Step 5:** After receiving the $Rep$ messages, each manager calculates the network-wide most beneficial replacement, say $Rep^*(r^*, v^*, i^*, k^*)$, the one of maximum relative gain, and updates its configuration matrix $\boldsymbol{H}$ correspondingly, setting $H_{v^*}^{k^*} = 0$ and $H_{v^*}^{i^*} = 1$. At this point only the configuration matrices of the managers are updated. Only after the completion of the algorithm managers fetch and cache new information items and replace cached ones (e.g. fetch item $i^*$ and replace item $k^*$).

**Step 6:** Repeat steps 1-5 until no further replacements are beneficial for the network, i.e. no positive relative gain exists.

The proposed algorithm is based on cooperative decision making. At each iteration the cache managers cooperate through appropriate message exchanges towards identifying the maximum relative gain. Thus, it can be shown that since any change performed in the cache configuration decreases the overall network traffic, the proposed algorithm finally converges to an equilibrium point where no further improvement is possible. We should mention here that it does not necessarily converge to the optimal cache assignment, but to a local minimum of the objective function given the initial cache configuration.

### B. Holistic cache management algorithm

The holistic algorithm is of similar nature and towards the same objective. Its distinguishing characteristic though is that each manager operates on its own by performing replacements on the respective cache. It can be thought of as a sequence of Gauss-Seidel iterations [23], where at each iteration a single cache manager, say $v \in \mathcal{V}$, autonomously decides and executes the following steps. Steps 1-3 are identical to the cooperative algorithm and are omitted:

**Step 4:** The replacement of maximum relative gain $r = g_i - l_k$ is performed by manager $v$. The rest of the cache managers are notified through the report message $Rep(r, v, i, k)$.

**Step 5:** After receiving the $Rep$ message every manager updates its configuration matrix $\boldsymbol{H}$ correspondingly, setting $H_v^k = 0$ and $H_v^i = 1$.

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

Besides, though the cache updates may be applied asynchronously among the nodes, we assume that only a single node may modify the cache configuration at a given time. This is due to the fact that each node needs to know the current cache configuration of the network, in order to calculate the gain and loss metrics. Thus, each modification is advertised to the rest cache managers. 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.

### C. Myopic cache management algorithm

In both the holistic and the cooperative algorithms every node needs to acquire global knowledge regarding the demand pattern for the decision making. However, in highly dynamic environments the amount of information that needs to be circulated among the managers becomes significant, causing thus non-negligible communication overhead. Even worse the

TABLE I
COMMUNICATION AND COMPUTATIONAL COMPLEXITIES.

| Complexity | Cooperative | Holistic | Myopic |
|---|---|---|---|
| Communication for initialization | $O(V^2M)$ | $O(V^2M)$ | 0 |
| Communication per iteration | $O(V^2)$ | $O(V)$ | $O(V)$ |
| Computational per iteration | $O(V^2M + V\log(M-C) + V\log C)$ | $O(VM + \log(M-C) + \log C)$ | $O(M + \log(M-C) + \log C)$ |

required information may not be available on time, making hence such an approach inapplicable.

For such scenarios we derive an alternative algorithm. We assume that each manager has no information about the demand patterns at the other caches, and thus makes decisions based only on local information. That is each node $v$ has to select his own cache configuration $\boldsymbol{H}_v^m$ for $m = 1 \dots M$, so as to minimize the traffic cost for the demand it serves. Thus, its objective function now becomes:

$$T_v(\boldsymbol{H}) = \sum_{m=1}^{M} r_v^m d_{u_m v}, \tag{3}$$

where $u_m$ the nearest cache hosting a 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.

## V. COMPLEXITY ANALYSIS

In this section we present the communication and computational complexity of the above algorithms. This will give significant insight regarding the incurred computational burden for each manager and the communications requirements regarding the substrate. Our analysis is performed regarding both the initialization and the per iteration complexity. An important parameter that affects both the total communication and the computational complexity of each approach is the number of iterations required for convergence to an equilibrium point. Since this is difficult to be calculated analytically we perform a characterization through simulations in section VI.

### A. Communication Complexity

We assume that each cache manager is aware of the initial cache configuration of the remote caches. Such information is available at contemporary CDNs [22] and Web caches (digest [24] or summary [25]) and could be easily provided by the ICN architecture.

At the initialization phase of both the cooperative and the holistic algorithms each cache manager needs to have a network-wide knowledge of the demand patterns. This requires each manager to forward its local demand vector to all the other cache managers. For this purpose a message $r_v$ (the demand vector at node $u$) of size $M$ needs to be forwarded to any other cache manager, leading thus to a total of $V(V-1)$ overhead messages for the initialization phase. As a result, the communication complexity of the initialization phase for both algorithms is $O(V^2M)$. On the other hand, the respective communication complexity of the myopic algorithm is zero, since decisions are made based only on local demand pattern, information that is available at manager level.

Regarding the amount of communication overhead induced per iteration, the cooperative algorithm, in order to calculate the maximum relative gain, requires a total of $O(V^2)$ messages. In particular, each manager needs to "broadcast" its $Rep$ message of length 4 in words. On the other hand, in the holistic and the myopic algorithm a single manager performs locally the cache updates and forwards to the rest of the managers a single $Rep$ message, for consistency purposes. This leads to a communication complexity per iteration of $O(V)$ messages. The multiplicative constant is 4 in all these cases (size of the $Rep$ message).

### B. Computational Complexity

In order to calculate the computational complexity of each algorithm we define the calculation of the traffic cost for node $u$ to access item $m$ stored at node $v$ (i.e. a single multiplication, $r_u^m d_{vu}$) as the basic operation. Thus, the complexity of each algorithm is calculated as the number of basic operations required.

The cooperative algorithm requires each manager at each iteration to perform $V \cdot M$ basic operations for the calculation of the $\boldsymbol{g}$ and the $\boldsymbol{l}$ vectors. In order to get the $g^*$ and the $l^*$ one max-heap and one min-heap operations are executed by each manager. The max-heap operation has a computational complexity of $O(\log(M-C))$, where $C$ the storage capacity of each node. Similarly the min-heap operation has a computational complexity of $O(\log C)$. So the computational complexity of the cooperative algorithm for each iteration is $O(V^2M + V\log(M-C) + V\log C)$.

The holistic algorithm is of the same computational complexity per manager. However, here only a single manager computes the relative gain within an iteration and not all of them. So the computational complexity per iteration of the holistic algorithm is $O(VM + \log(M-C) + \log C)$.

The myopic algorithm requires for the calculation of the $\boldsymbol{g}$ and the $\boldsymbol{l}$ vectors only $M$ constant time operations, since only the local demand pattern is known to each manager. So the computational complexity of the myopic algorithm for each iteration is $O(M + \log(M-C) + \log C)$. Table I summarizes the communication and computational complexity of the proposed distributed on-line cache management algorithms.

## VI. PERFORMANCE EVALUATION

In this section, we evaluate through simulations the performance of the proposed cache management algorithms. For comparison purposes we simulate also a *greedy* centralized cache assignment algorithm and a *local* selfish approach. In the latter each manager has no information regarding neither the remote request patterns nor the assignment of the other caches, and thus makes decisions based only on local information.

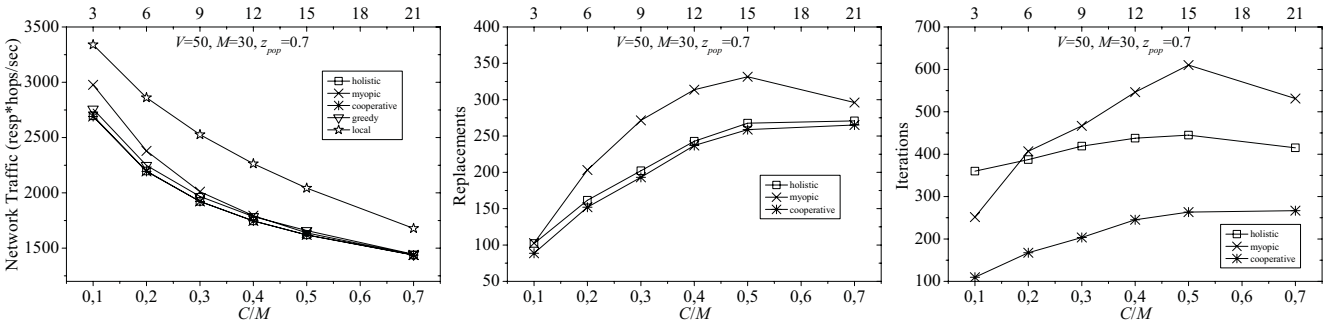


Fig. 2. The performance of the proposed cache management algorithms vs. the fraction ($p = C/M$) of the items that can be stored in a cache, when the variable is the capacity $C$ of each cache. The secondary $x$-axis is the actual capacity $C$ of each cache.

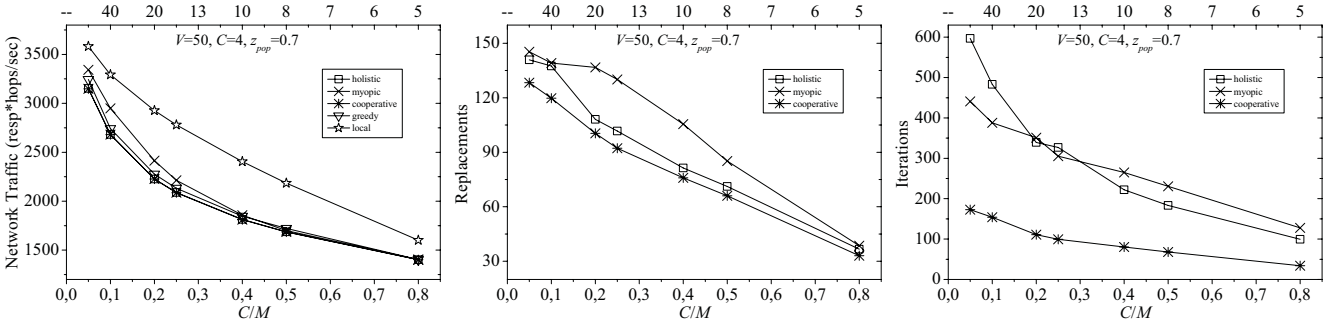


Fig. 3. The performance of the proposed cache management algorithms vs. the fraction ($p = C/M$) of the items that can be stored in a cache, when the variable is the number of information items $M$ in the network. The secondary $x$-axis is the actual number of information items $M$ in the network.

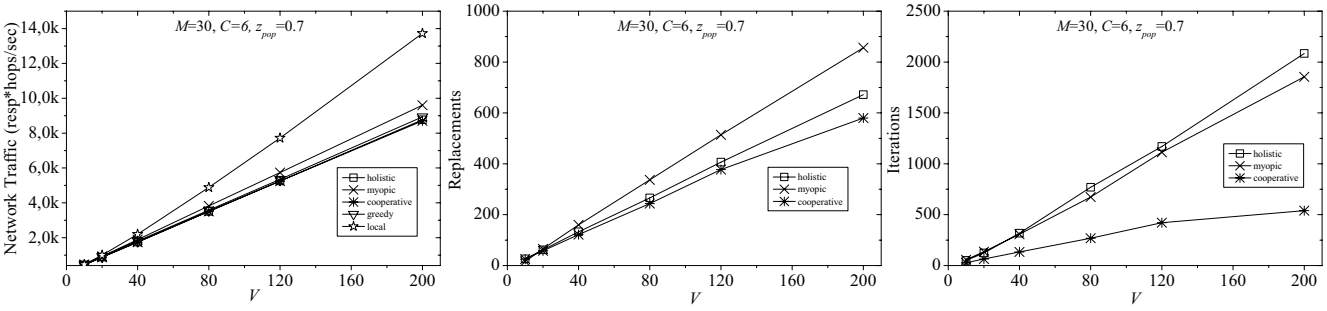


Fig. 4. The performance of the proposed cache management algorithms vs. the number $V$ of caches in the network.

Actually, the $C$ locally most popular items are cached. The greedy algorithm, reported in [26], is an iterative but off-line centralized algorithm which requires $V^2M$ iterations. It gives solutions of high quality, since its median performance is within a factor of 1.1 - 1.5 of the optimal and around a factor of 4 for the maximum cases.

Each point of the following figures is the mean value out of fifty executions starting from different initial cache assignments and for fifty different network topologies. In particular, we use the Waxman model [27] to generate the network topologies, where a pair of nodes $\{u, v\}$ is connected with probability $p_{uv} = \beta \mathrm{e}^{\frac{-d_{uv}}{L\alpha}}$ depending on their distance $d_{uv}$, with $L$ being the maximum distance of any two nodes and $\alpha, \beta \in (0, 1]$. Parameter $\beta$ controls the density of the graph (the larger the value of $\beta$ the denser is the graph), while $\alpha$ controls the connectivity of the graph (the smaller the value of $\alpha$ the larger is the number of short edges) [28]. Our fifty random network topologies arise from random values of parameters $\alpha$ and $\beta$.

Information-centric research lacks publicly available data sets for meaningful evaluation. Thus, synthetic workload generation according to realistic assumptions is widely accepted. Each information item is characterized by two parameters, namely *popularity* and *locality*. Popularity refers to the request rate related to an item and locality to the region of the topology likely to originate requests. We denote by $p^m$ (respectively $\ell^m$) the popularity (respectively the locality) of item $m$. Popularity and locality values are computed according to a Zipf law of exponents $z_{pop}$ and $z_{loc}$ respectively. Requests are issued from a set of nodes computed using $\ell^m$. In particular, $\lceil \ell^m V \rceil$ nodes are potential issuers of requests related to item $m$. This set of nodes is computed by choosing randomly a central node and its $\lceil \ell^m V \rceil - 1$ closest nodes, by executing a Breadth First Search algorithm.

Our figures depict for each cache management algorithm the following performance metrics:

- The *overall network traffic, NT* (in $reqs \cdot hops/sec$) at the equilibrium.

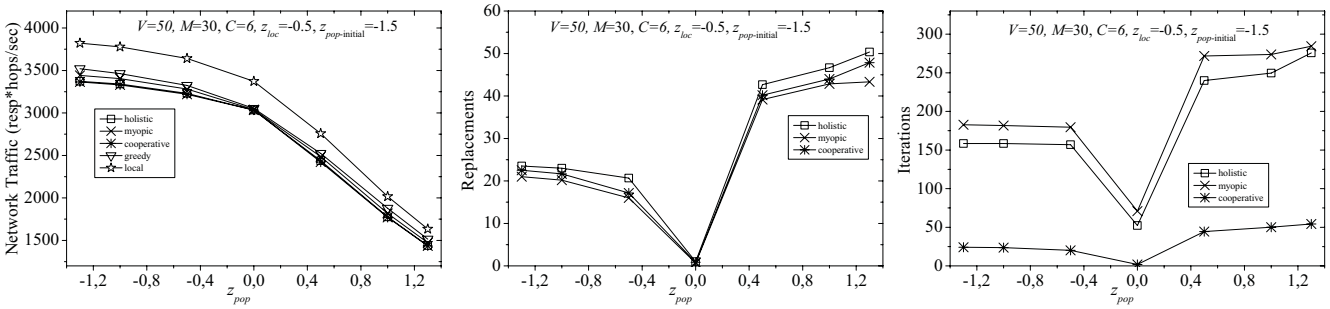


Fig. 5. The performance of the proposed cache management algorithms vs. the popularity exponent of the items $z_{pop}$. The initial cache assignment is the output of the "greedy" algorithm for $z_{pop} = -1.5$ and $z_{loc} = -0.5$.

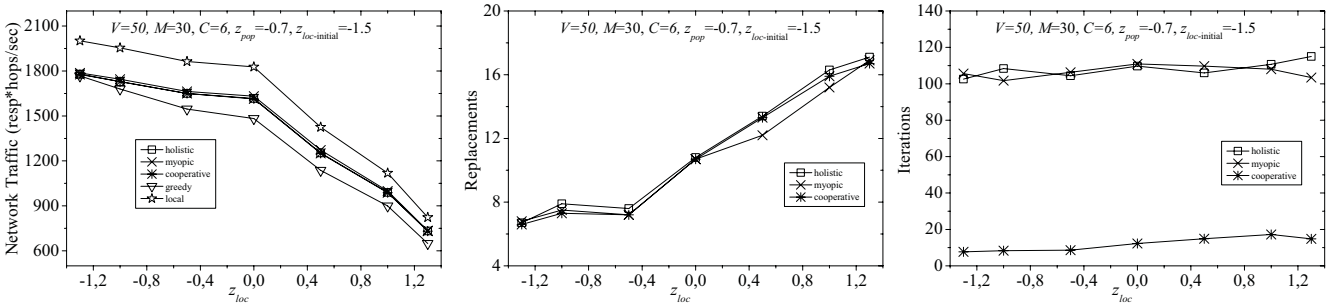


Fig. 6. The performance of the proposed cache management algorithms vs. the locality exponent of the items $z_{loc}$. The initial cache assignment is the output of the "greedy" algorithm for $z_{loc} = -1.5$ and $z_{pop} = -0.7$.

- The *total number of replacements, RE* after the completion of the cache management algorithms (the actual number of items fetched by the managers so as to update their caches).
- The *total number of iterations, IT* required for convergence and is indicative of the running time of each algorithm. *IT* multiplied by the communication complexity derived at section V-A provides the communication cost, while multiplication with the computational complexity calculated in section V-B reveals the computational cost of each algorithm.

Initially, we assume uniform locality and popularity for each item. Uniform locality implies that requests are generated from every node in the network for every item with the same probability i.e. the neighborhood of interest for each item is the whole network. Uniform popularity implies that the request rate generated from each node $y$ of the network for item $m$ is $r_u^m$ and it is the same for all nodes for that specific item $m$ ($r_u^m = r^m$, $\forall u \in V$). This request rate varies from 0 - 250 requests/sec.

In this setting, we depict in Fig. 2 and Fig. 3 the impact of cache capacity, expressed as the fraction of the items that can be stored in a cache ($p = C/M$). In the former the variable is the actual capacity $C$ of each cache, while in the latter the number of information items $M$ in the network. Regarding the *NT* metric we notice similar behaviour for both cases, with our nework-wide approaches having performing better than the greedy algorithm, but as expected with the benefit diminishing as we relax the storage capacity constraint and allowing more items to fit in each cache. However, the complexity related metrics exhibit different behaviour. In particular, while *RE* and *IT* are increasing in the capacity $C$ of each cache, they are decreasing as the number of items $M$ decreases. This is justified by the fact that as the available capacity increases more beneficial replacements appear as candidates, while the decreasing number of items has the opposite impact.

In Fig. 4 we examine the impact of the number of cache locations $V$ in the network. We notice that all the performance metrics exhibit a linear behaviour of slope $V$, indicating a per cache identical performance that is justified due to the uniform popularity and locality parameters. Thus, for the uniform scenario we notice that the network wide approaches outperform significantly the local ones but also the greedy one. Regarding the communication/computational complexity the proposed cache management algorithms perform significantly better than the centralized greedy, which requires $V^2M$ iterations. Moreover the cooperative algorithm requires up to ten times less iterations than the other two algorithms but its communication complexity is $V$ times larger than the communication complexity of the other two algorithms, while its computational complexity is $V$ times larger than the computational complexity of the holistic algorithm and $V^2$ times larger than the computational complexity of the myopic algorithm. The cooperative and the holistic algorithms perform better regarding the number of iterations (convergence time) due to the fact that all the actions are either explicitly coordinated (cooperative) or implicitly through the common objective (holistic). On the other hand, the local approaches suffer from the probably counteracting objectives of the individuals. So in general fast convergence comes with a higher communication and computational cost.

In our second set of experiments we investigate how adap-

tive our algorithms are as either popularity or locality changes. Using as initial cache assignment that of the off-line greedy algorithm for given locality and popularity values, we depict the iterations and replacements required by each algorithm to adapt to the new environmental parameters. In Fig. 5 we see that as the item popularity becomes more uniform (near zero exponent) less replacements are required since all the assignments are of almost equal performance. This is also evident from the network traffic plots, where the performance gap of the proposed approaches diminishes.

In Fig. 6 we examine the impact of locality variations on performance. We notice that changes of the locality exponent cause a domino effect requiring significant reorganization of the cache contents, since they alter the topology of the demands of the network under consideration. This justifies also the better performance of the off-line greedy algorithm, which selects the items assigned from scratch, at the cost of increased complexity.

Fig. 5 and Fig. 6 could also be used as a benchmark for the cache managers in their decision to reassign or not the cached items upon the detection of a change in the popularity or the locality pattern. Particularly the difference between the network traffic cost of the initial cache assignment and the traffic cost after the completion of the algorithms combined with the communication and computational complexity enables the cache managers to skip or not the cache reassignment. For example when the observed $z_{pop}$ alters from $z_{pop} = -1.5$ to $z_{pop} = 0$ (Fig. 5) we observe only a $10\%$ decrease on the overall network traffic, whereas when the $z_{pop}$ alters to $z_{pop} = 1.3$ the decrease in the overall network traffic is more than $60\%$, meaning that in the first case the managers could skip the reassignment of the caches, whereas in the second case the reassignment is crucial at almost the same cost with the first case.

## VII. Conclusion

In this paper, we proposed an autonomic cache management architecture that dynamically reassigns information items to caches in an ICN. The reassignment decisions are based on the real time observed item request patterns such as their popularity and locality and not in static off-line predictions. Particularly we presented three distributed on-line cache management algorithms which require different level of cooperation between the autonomic managers and we compare them against their performance, complexity, message overhead and convergence time.

It is evident that the network wide knowledge and cooperation give significant performance benefits and reduce significantly the time to convergence, but at the cost of additional message exchanges and computational effort. 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 caches of the network, as well as the processing load of each cache when assigning items to them. Finally a testbed implementation and evaluation (PURSUIT testbed [2]) are in the imminent plans.

## Acknowledgment

V. Sourlas' and L. Gkatzikis' work is co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning of the National Strategic Reference Framework (NSRF) Research Funding Program: Heracleitus II - Investing in knowledge society through the European Social Fund. This work has also been supported by the European Commission through the FP7 PURSUIT program, under contract ICT-2010-257217.

## References

[1] Named Data Networking project, available at http://named-data.net, 2011.
[2] PURSUIT project, available at http://www.fp7-pursuit.eu, 2011.
[3] B. Ohlman et al, "First NetInf architecture description," Apr. 2009. http://www.4ward-project.eu/index.php?s=file_download&id=39.
[4] J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," In www.research.ibm.com/autonomic/research/papers/, 2003.
[5] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M.O. Foghlu, W. Donnelly and J. Strassner, "Towards Autonomic Management of Communications Networks," IEEE Communications Magazine, Vol. 45, No. 10, pp. 112-121, 2007.
[6] N. Tcholtchev, M. Grajzer and B. Vidalenc, 'Towards a Unified Architecture for Resilience, Survivability and Autonomic Fault-Management for Self-managing Networks," In ICSOC/ServiceWave Workshops, 2009.
[7] The EU FP7 AutoI Project, http:ist-autoi.eu.
[8] D. Clark, C. Partridge, J. Ramming and J. Wroclawski, "A Knowledge Plane for the Internet," IN ACM Sigcomm, Karlsruhe, Germany, 2003.
[9] L. Qiu, V.N. Padmanabhan, G. Voelker, "On the placement of web server replicas," In IEEE INFOCOM, Anchorage, USA, Apr. 2001.
[10] 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.
[11] E. Cronin, S. Jamin, C. Jin, T. Kurc, D. Raz, Y. Shavitt, "Constrained mirror placement on the Internet," IEEE JSAC, 36(2), Sept. 2002.
[12] I.D. Baev, R. Rajaraman, C. Swamy, "Approximation algorithms for data placement problems," SIAM J. Computing, vol. 38, 2008.
[13] D. Applegate, A. Archer, V. Gopalakrishnan, "Optimal content placement for a large-scale VoD system," In ACM CoNEXT, Philadelphia, USA, Dec. 2010.
[14] P. Rodriguez, C. Spanner, E.W.Biersack, "Analysis of Web Caching Architectures: Hierarchical and Distributed Caching," ACM Trans. on Networking, Aug. 2001.
[15] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed selfish replication," IEEE Trans. Parallel Distrib. Syst., vol. 17, no. 12, pp. 14011413, 2006.
[16] S. Zaman, D. Grosu, "A Distributed Algorithm for the Replica Placement Problem," IEEE Trans. Parallel and Distrib. Syst., Jan. 2011.
[17] S. Borst, V. Gupta, A. Walid, "Distributed Caching Algorithms for Content Distribution Networks", in IEEE INOFCOM, San Diego, USA, March 2010.
[18] V. Sourlas, P. Flegkas, G. S. Paschos, D. Katsaros, and L. Tassiulas, "Storage planning and replica assignment in content-centric publish/subscribe networks," Comput. Netw. (2011), doi:10.1016/j.comnet.2011.07.023
[19] 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.
[20] G. Carofiglio, M. Gallo and L. Muscariello, "Bandwidth and storage sharing performance in information-centric networking," in ACM SIGCOMM workshop on information-centric networking, Toronto, Canada, 2011.
[21] M. Sayal, Y. Breitbart, P. Scheuermann, R. Vingralek, "Selection algorithms for replicated web servers," ACM Performance Evaluation Rev. 26 (3) 1998.
[22] J. Kangasharju, J. Roberts, K. Ross, "Object replication strategies in content distribution networks," Comput. Commun. 25 (4) (2002) pp. 376-383.

[23] 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.
[24] A. Rouskov, D. Wessels, "Cache Digest," In 3rd Inter. WWW caching workshop, June 1998.
[25] 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.
[26] L. Qiu, V. Padmanabhan, G. Voelker, "On the placement of web server replicas," In Proc. of the IEEE INFOCOM, 2001, pp. 15871596.
[27] B. Waxman, "Routing of multipoint connections," IEEE JSAC vol. 6, pp. 16171622, 1998.
[28] P. Habegger, H. Bieri, "Modeling the topology of the Internet: an assessment," Technical Report, Institut fr Informatik und angewandte Mathematik, University of Bern, Switzerland, IM-02-2002, 2002.