Computer Networks 55 (2011) 4021–4032

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Storage planning and replica assignment in content-centric publish/subscribe networks ☆,☆☆

Vasilis Sourlas [a,*], Paris Flegkas [a,b], Georgios S. Paschos [a,b], Dimitrios Katsaros [a,b], Leandros Tassiulas [a,b]

[a] Department of Computer & Communication Engineering, University of Thessaly, Greece
[b] Centre for Research & Technology Hellas (CERTH-ITI), Greece

ARTICLE INFO

Article history:
Received 30 November 2010
Received in revised form 4 May 2011
Accepted 8 July 2011
Available online 2 August 2011

Keywords:
Storage planning
Replica assignment
Content-centric publish/subscribe networks

ABSTRACT

Content-centric publish/subscribe networking is a flexible communication model that meets the requirements of the content distribution in the Internet, where information needs to be addressed by semantic attributes rather than origin and destination identities. In current implementations of publish/subscribe networks, messages are not stored and only active subscribers receive published messages. However, in a dynamic scenario, where users join the network at various instances, a user may be interested in content published before its subscription time. In this paper, we introduce a mechanism that enables storing in such networks, while maintaining the main principle of loose-coupled and asynchronous communication. Furthermore, we propose a new storage placement and replica assignment algorithm which differentiates classes of content based on their popularity and minimizes the clients response latency and the overall traffic of the network. We also present and compare two replica assignment alternatives and examine their performance when both the locality and the popularity of users request change. The performance of our proposed placement and replica assignment algorithm and the proposed storing mechanism is evaluated via simulations and insights are given for future work. The proposed mechanism is compared with mechanisms from the CDN (Content Delivery Networks) context and performs as close as 1–15% (depending on the conducted experiment) to a greedy (near optimal) approach installing up to 3 times less storage servers in the network and providing the necessary differentiation among the classes of the content.

1. Introduction

Publish/subscribe (pub/sub) systems (topic based or content based) are organized as a collection of autonomous components, namely the clients and the event dispatchers. Clients act either as publishers, publishing new events in the network, or as subscribers by subscribing to the classes of events they are interested in. The event dispatchers (or event brokers or simply brokers) on the other hand, are responsible for collecting subscriptions and forwarding publications to interested subscribers. In pub/sub networks, the selection of a message is determined entirely by the client, which uses expressions (filters) that allow sophisticated matching on the event content.

☆ V. Sourlas' 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.
☆☆ An initial approach on storing/replication in topic based pub/sub networks is presented in [11].
* Corresponding author.
E-mail addresses: vsourlas@inf.uth.gr (V. Sourlas), pflegkas@uth.gr (P. Flegkas), gpasxos@uth.gr (G.S. Paschos), dkatsar@uth.gr (D. Katsaros), leandros@uth.gr (L. Tassiulas).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.07.023

In current pub/sub implementations, any event is guaranteed to reach all interested subscribers as long as their subscriptions are known to the network at publish time, assuming stable topology and no queuing overflows. However, in a dynamic distributed environment, clients join and leave the network over time, and it is possible that a subscriber joins the network after the publication of an interesting message. In current pub/sub systems, it is not possible for a new subscriber to retrieve already published messages that match his/her subscription. Therefore, enabling the retrieval of past published content by means of storing is one of the most challenging problems in pub/sub networks.

Content delivery servers ("surrogate servers" in CDNs or simply "stores" in this work) replicate the whole content of a given server and target to speeding up the delivery of content by reducing the load on the origin servers and the network itself. When a client is interested in a piece of information of a given server, his/her request is redirected to one of the existing stores (e.g., the closest one or the one satisfying other criteria such as the load of the candidate store). Since stores serve only a portion of the total requests and are placed closer to the client, clients are served faster. A client's request is redirected to a store only if that store is a replica of the targeted server, otherwise the request is directed and served by the server itself.

In this work, we assume that messages are classified according to their class (*topic*) and we:

- Enhance the pub/sub communication paradigm with an advertisement and a request/response mechanism so that stores can advertise the class of the content they have stored and clients can retrieve it.
- Propose a new algorithm for the selection of M storage points among the N brokers of the network ($M < N$) based on: (a) the locality and the popularity of the interests for each topic, (b) the targeted replication degree of each topic (as replication degree we name the number of replicas k_t ($1 \leqslant k_t \leqslant M$) of the topic $t \in T$ among the stores, which is analogous to its popularity) and (c) the storage capacity (limitation) L of each store.
- Propose two alternative mechanisms for the assignment of the replicas of each topic $t \in T$ among the selected stores.
- Evaluate through simulations our design of the storing mechanism and the new placement and replica assignment algorithm.

The objective of our scheme is to minimize the total traffic load of all the classes of content in the network subject to installing the minimum number of stores and given that stores have storage limitations.

The rest of the paper is organized as follows. In Section 2 a brief related work on storing in pub/sub architectures is given, followed by a description of the storage placement problem while, in Section 3 we describe the problem under investigation. In Section 4, we shortly describe the pub/sub architecture and present the proposed advertisement and request/response mechanism. The new algorithm for the selection of the stores' location and the replica assignment of the content is presented in Section 5. Section 6 is devoted to performance evaluation via simulations. Finally, we conclude the paper and give insights for future work in Section 7.

2. Related work

Internet's usage has considerably changed over the past years from a resource sharing mechanism between a pair of hosts to a content distribution and retrieval mechanism. In that changing environment the pub/sub paradigm is becoming increasingly popular for content access and dissemination. Particularly there are several research efforts that develop an overlay event notification service like IBM's Gryphon [1], Siena [2], Elvin [3] and REDS [4] which implement the pub/sub architecture. Moreover, there are also several research efforts aiming to switch from host-oriented to content-oriented networking like CCN [5], DONA [6], PURSUIT [7] and SAIL [8], which attempt to name data/content instead of naming hosts in order to achieve scalability, security and performance.

The various pub/sub models are classified according to the semantics of the subscription language. Among the pub/sub models the most known are the *topic-based* pub/sub, which enables information consumers to register according to a set of predefined topics organized into a hierarchy, and the *content-based* pub/sub, which supports subscriptions that follow an attribute/value scheme.

In the area of caching/storing and replication in content-centric pub/sub networks in [9] a historic data retrieval pub/sub system is proposed, where databases are connected to various brokers, each associated with a topic to store. In [9] every message is stored only once and no placement strategies have been examined, while there is no description of the mechanism for the retrieval of the stored data. Moreover, in [10] we introduced an opportunistic caching scheme for pub/sub networks where each broker of the network is a potential caching point, while a first attempt with an off-line replication algorithm in topic-based pub/sub networks is presented in [11].

On the other hand the placement problem, in the context of CDNs, is a thoroughly investigated problem. Particularly in [12,13] authors approached the placement problem with the assumption that the underlying network topologies are trees. This simple approach allows the authors to develop optimal algorithms, but they consider the problem of placing replicas for only one *origin server*. The placement problem is in fact an NP-hard problem [14], but there are a number of studies [15–20] 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 N nodes targeting the minimization of a given objective. In some cases, it can be shown that this problem reduces to the well known *k-median* problem.

More placement algorithms have been proposed in [14]. Particularly, authors formulate the problem as a combinatorial optimization problem, prove that this is NP-hard and develop and compare four natural heuristics algorithms. They found that the best results are obtained with heuristics that have all the stores cooperating in making the

replication decisions. Finally, in [21] authors introduce a framework for evaluating placement algorithms. They classify and qualitatively compare placement algorithms using a generic set of primitives that capture several objective and near optimal solutions. While most models have a similar cost function (optimize bandwidth and/or storage usage costs for a given request pattern), less attention has been given to network-wide constraints (limited storage capacity of the stores).

3. Problem description

In our work, we assume a content-centric pub/sub network with arbitrary topology of N nodes. T different topics should be stored at M stores, where each store has the capability to store L topics. Each topic $t \in T$ should be replicated k_t times. Requests for the topics are generated at various nodes and they trigger the transfer of the requested item from a store to the node where the request was generated. The proposed mechanism is composed by two phases, typical in any network management task, namely the *Planning* and the *Assignment* phase. In the *Planning* phase, the proposed mechanism selects M points out of the N nodes of the network to place the stores, while in the *Assignment* phase, each topic t is assigned at exactly k_t different stores with the target to minimize the total traffic load in the network.

Generally, in a real content delivery pub/sub network the *Planning* of stores changes rarely, since it requires the reallocation of the stores among the network nodes. On the other hand, the *Assignment* of the topics is more flexible and the CDN provider is able to reassign the topics among the stores when the locality and the popularity patterns change is such a way that the performance of the network is degraded with the existing configuration. Of course, a reassignment requires the calculation of the new places for each topic and the transfer of topics to those locations, but as shown later in the performance analysis, it is an efficient way to maintain the performance of the system at high levels without re-planning the whole CDN network.

The storage capacity of each replication server usually refers to TBytes but for simplicity we assume here that the number of messages is the same for each topic and messages are of the same size, leaving for future work the case of different sizes. Alternatively, at each snapshot of the system in the network, there exists the same number of messages for each topic. The L parameter is a limitation introduced by the storage providers and refers to the maximum storing capability of each store in the network. On the other hand, the k_t parameter (replication degree of each topic) is a limitation introduced by the storage provider and refers to the number of replicas that the content provider is willing to pay for. Finally, the M parameter refers to the number of stores that a storage provider should install in the network to serve the storage demands of the topics.

4. Enabling storing in pub/sub networks

In this work, we consider a pub/sub network which uses the subscription forwarding routing strategy [2], where the routing paths for the published messages are set by the subscriptions, which are propagated throughout the network so as to form a tree that connects the subscribers to all the brokers in the network. In that scheme, publishers join the network when they have something to publish, therefore in our approach the entity of the origin server does not exist.

In a pub/sub network, when a client issues a subscription, a message containing the subscription filter is sent to the broker the client is attached to. The filter is inserted in a Subscription Table (`ST`), together with the identifier of the subscriber. Then, the subscription is propagated by the broker, which now behaves as a subscriber with respect to the rest of the dispatching network, to all of its neighboring brokers. In turn, the neighbors record the subscription and re-propagate it towards all further neighboring brokers, except for the one that sent it. Finally, each broker in the network has a `ST`, in which for every neighboring broker there is an associated set of filters containing the subscriptions issued by them.

4.1. Advertisement and request/response mechanism

In this section, we present the advertisement and the request/response mechanism, which provides a pub/sub system with the ability to store and retrieve information published in the past and make it available for future clients. Particularly, we will present the new mechanisms through the example of Figs. 1 and 2.

In order to retrieve stored information, we enhance the system with three additional types of messages (besides the already existing `Publish ()` and `Subscribe ()`), `Advertise ()`, `Request ()` and `Response ()`. We also add to the system a new feature called `Advertisement Table (AT)`, similar to `ST`, which is used to store advertisements. When a new store "*str1*" is installed at broker 6 (Fig. 1), it issues a `Subscribe ()` message with the topics (class of events) that is willing to store (*top_a* and *top_b* in the given example). In that way, it acts as a client to future publications matching the subscribed topics and, each time a relevant publication occurs (i.e. publisher attached to broker 1 publishes message *msg_a* matching *top_a*), it stores the message (the message is also stored to "*str2*"). The "*str1*" also issues an `Advertise ()` message, which contains the topics that stores and the distance in hops from the store (the distance attribute is built hop by hop). Advertisements are treated similarly to subscriptions and form a tree that connects the "*str1*" to all the brokers in the network. Advertisements are inserted in the (`AT`). Coverage also occurs with advertisements, as with subscriptions, but in a slightly different way. Particularly, when a broker receives an advertisement, checks in the distance field and if the distance is equal to another entry (for the same topics), it adds the advertisement to the `AT` and stops forwarding the advertisement. Keeping more than one entry for the same topic in an `AT`, enables load balancing capabilities to requests passing from that particular broker. On the other hand, when a broker receives an advertisement for a store which is closer compared to the other stores already in the `AT`, it adds the advertisement to the `AT`, removes the previous entries and forwards it further (brokers 5 and 6 in Fig. 1). Finally, when

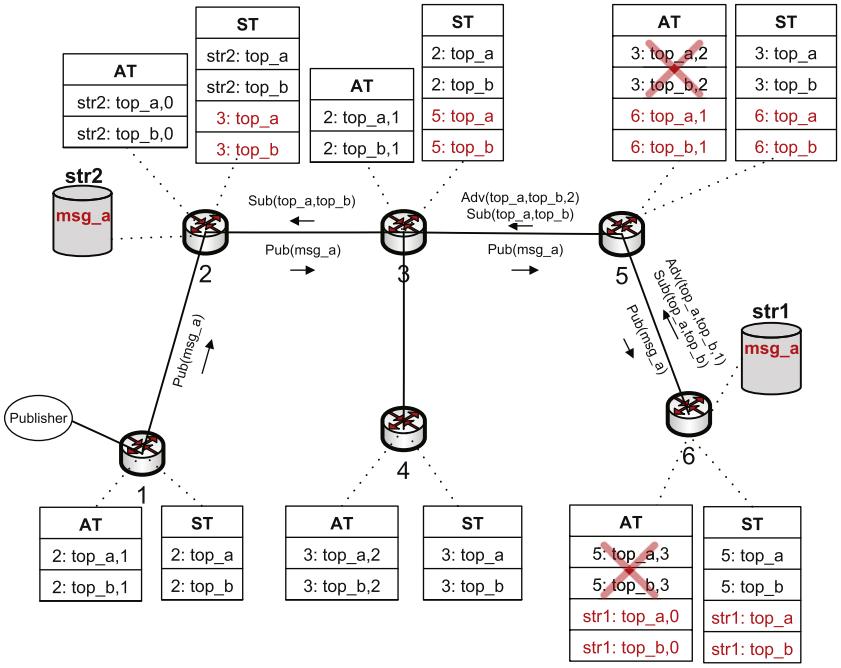

Fig. 1. Advertising and Storing of information (in red are the new entries of STs and ATs created by the installation of the "str1"). Publisher at broker 1 publishes a message *msg_a* that matches *top_a* and is stored at "str1" and "str2". "2:top_a,1" in AT means that any request for topic *a* should be forwarded to broker 2 and the closest store for topic *a* is 1 hop away. "2:top_a" in ST means that any publication matching topic *a* should be forwarded to broker 2. (For interpretation of color mentioned in this figure the reader is referred to the web version of the article.)

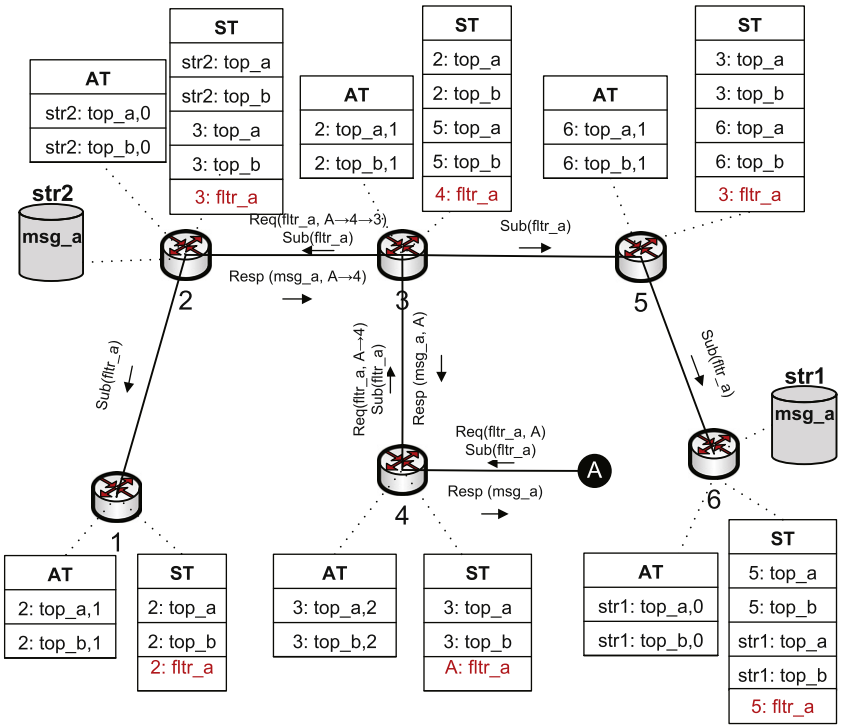

Fig. 2. Retrieval of stored information using the request/response mechanism (in red are the new entries of STs created by the subscription of client *A*). (For interpretation of color mentioned in this figure the reader is referred to the web version of the article.)

a broker receives an advertisement for a store which is further compared to the other stores already in the AT, it simply stops the forwarding of the advertisement without changing the entries of the AT (broker 3 in Fig. 1).

When a client (client *A* in Fig. 2), is interested in retrieving stored content apart from subscribing (if he/she is also interested for future publications) he/she issues a request by sending a `Request ()` message containing the interested filter (*fltr_a*). Filters contains, apart from the topic that the client is interested in, more attributes to enable sophisticated matching. Source routing is used for the forwarding of the `Request ()` (the path is being built hop by hop and is included in the `Request ()` header). Broker 4 upon receiving the `Request ()` message checks in its (`AT`) for entries matching the requested topic (*top_a* in this case). The broker forwards the `Request ()` message to the broker who had advertised the matching topic and is closer to the client (in this example broker 3 and finally broker 2). Finally, "str2" receives the `Request ()` message, matches its stored content with the whole filter (not just the topic) and initiates a `Response ()` message for each match (messages *msg_a* in Fig. 2).

A `Response ()` message carries a stored message as well as the sequence of nodes carried by the initiating `Request ()` message (source routing). When a broker receives a `Response ()` message it pops off its identifier from that sequence and forwards it to the first broker of the remaining sequence. In the end, client *A* will receive every stored message matching his/her request.

5. Placement and replica assignment strategy

We use as the base of our placement and replica assignment scheme, algorithms presented in the context of CDN networks. Particularly in [14,15], authors developed several placement algorithms that use workload information, such as latency (distance from the storage points) and request rates, to make the placement decision. Their main conclusion is that the so called "*greedy*" algorithm that places stores based upon both a distance metric and request load, performs the best and is very close to the optimal solution.

5.1. Greedy algorithm

Here, we briefly present the greedy algorithm assuming that there exists only one class of content in our system (one topic), or equivalently there is no distinction in the content. We let r_i be the demand (in reqs/s) from clients attached to node i. We also let p_{ij} be the percentage of the overall request demand accessing the target server j (traditional placement algorithms replicate a specific origin server) that passes through node i. Also we denote the propagation delay (hops) from node i to the target server j as d_{ij}. If a store is placed at node i we define the Gain to be $g_{ij} = p_{ij} \cdot d_{ij}$. This means that p_{ij} percentage of the traffic would not need to traverse the distance from node i to server j decreasing the overall network traffic by:

$$d_{ij} \cdot \sum_{l=1}^{N} R_l,$$

where

$$R_l = \begin{cases} r_l & \text{if } i \text{ is on the path from } l \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$$

The greedy algorithm chooses one store at a time (we need k stores out of the N nodes of the network). In the first round, it evaluates each of the N nodes to determine its suitability to become a store (replication point of server j). It computes the Gain associated with each node and selects the one that maximizes the Gain. In the second round, searches for a second store which, in conjunction with the store already picked, yields the highest Gain. The greedy algorithm iterates until k stores have been chosen to replicate server j.

5.2. Modified greedy algorithm

In the pub/sub network architecture that we assume in this paper, the notion of an origin server – which is vital for the greedy algorithm – does not exist. Publishers join the network, publish their content and disappear. So in order to obtain the location of the stores we modify the greedy algorithm. Particularly we repeat the above procedure N times assuming each time that the targeted server j is a different node (broker) of the network. We get in that way N vectors of k possible stores. Precisely, each vector has N elements with k ones in the index of the selected stores and $N - k$ zeros in every other place. For example, vector [0 0 0 1 0 1] means that of the 6 nodes of the network the selected $k = 2$ possible stores are nodes 4 and 6. Finally, we select as our stores those k nodes that appeared more times in the per element summation of the N vectors and install at each one a store following the mechanism described in Section 4.1. The modified greedy algorithm presented here assumes uniform distribution of the probability among the N nodes of the network that publications could occur. Of course other forms of probability distributions could be used, and each vector should be first weighted with its probability before the per element summation of the N vectors.

5.3. Placement and replica assignment algorithm for pub/sub networks

Here, we use the modified greedy algorithm described above for the case where in our network exist T different classes of content (topics). Next, we present the Steps of the proposed algorithm side by side with the example of Fig. 3 (Table 1 contains all the useful parameters required by the proposed algorithm):

1. For each topic $t \in T$ we execute the modified greedy algorithm presented in Section 5.2 and we get T vector of possible stores $\mathbf{s}_t$. Regarding the example we get:
   $\mathbf{s}_a$ = [0 3 5 0 2 2]
   $\mathbf{s}_b$ = [0 2 5 0 5 0]
   $\mathbf{s}_c$ = [0 2 5 0 5 0]
   for the three topics accordingly. The [0 3 5 0 2 2] means that out of the $N = 6$ executions of the modified greedy algorithm, node 2 appeared 3 times, node 3 appeared 5 times and so on.
2. Each vector ($\mathbf{s}_t$) is weighted by $w_t = \frac{\sum_{i=1}^{N} r_i^t}{\sum_{i=1}^{N}\sum_{t=1}^{T} r_i^t}$. w_t shows the significance (popularity) regarding the traffic demand of each topic in the network. The weights for

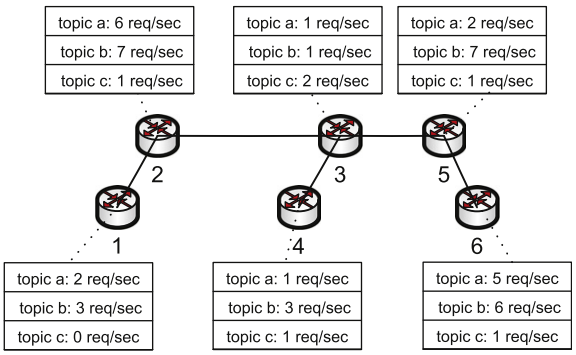

Fig. 3. Topology and workload information (requests/s) per each class of content ($T = 3$ topics) together with, $k_t = k = 2$, $L = 2$ and $M = 3$ form the inputs of the placement algorithm for the pub/sub network.

Table 1
Parameters used by the placement algorithm and its assignment alternatives.

r_i^t	: request rate for topic $t \in T$ in broker i
N	: number of nodes (brokers) in the network
M	: ($M < N$) number of stores in the network
k_t	: ($k_t \leqslant M$) replication of each topic $t \in T$ in the network
L	: storage capacity of each store in the network
T	: number of classes of content (topics)
w_t	: weight of topic $t \in T$ in the network
$\mathcal{S}$	: storage brokers vector
$\mathbf{s}_t$	: possible stores vector for topic $t \in T$
ξ_t	: relative weight of topic $t \in T$

the given example are:
$w_a = 17/50 = 0.34$, $w_b = 27/50 = 0.54$, $w_c = 6/50 = 0.12$
We obtain the following weighted vectors.

$w_a \cdot \mathbf{s}_a = [0\ 1.02\ 1.7\ 0\ 0.68\ 0.68]$
$w_b \cdot \mathbf{s}_b = [0\ 1.08\ 2.7\ 0\ 2.7\ 0]$
$w_c \cdot \mathbf{s}_c = [0\ 0.24\ 0.6\ 0\ 0.6\ 0]$.

3. We select as our stores those M nodes that appeared more times in the per element weighted summation of the T vectors. We call that vector the *storage brokers vector* $\mathcal{S}$. The per element summation of the above three vectors into a single vector gives [0 2.34 5 0 3.98 0.68] meaning that the final $M = 3$ stores in $\mathcal{S}$ are nodes 3, 5 and 2.
4. For each topic t, starting from the most significant (based on the weight), we assign k_t stores following the procedure below:
   - For each entry in the $\mathbf{s}_t$ of topic t calculated in step 1 assign a store if that entry also appears in the $\mathcal{S}$, calculated in step 3, and only if in that store has been assigned less than L (storage capacity) topics until we get k_t stores (replication of topic t).

In the example starting from topic b then topic a and finally topic c (based on their weights) we assign them to $k = 2$ stores. Topic b is assigned to nodes 3 and 5 which were the nodes for topic b appeared more times in Step 1. Topic a is also assigned to nodes that were produced by Step 1, nodes 2 and 3, while topic c is assigned to nodes 2 and 5. Node 5 was among the most popular selections produced by Step 1 while node 2 was the only store in $\mathcal{S}$ with less than $L = 2$ assignments.

Step 4 of our algorithm is also known as the *Generalized assignment problem* which even in its simplest form is reduced to the NP-complete multiple knapsack problem. In this paper, for the solution of the assignment problem we used the heuristic approaches described above and in Section 5.5, while more approaches could be found in literature [22].

5.4. Cost model

Steps 1–3 of the proposed algorithm described above comprise the *Planning* phase of the algorithm while Step 4 is the *Assignment* phase. In this section, we present the cost model of the *Assignment* phase, which as mentioned above is an NP-complete problem. The access of an information item stored in store x by node y generates a traffic load equal to the length (number of hops) of the path from x to y. Given that we wish to optimize the total traffic load, the access scheme is that we always access the closest store (shortest path) among those holding the specific item. Thus given the access mechanism, we seek to decide the replica assignment of each topic.

Let C be the traffic load corresponding to any storage configuration.

For that storage configuration we can write:

$$C = \sum_{t=1}^{T} C_t, \tag{1}$$

where C_t is the traffic load corresponding to configuration of topic t only.

We then have,

$$C_t = \sum_{n=1}^{k_t} \sum_{l \in \mathcal{N}_n^t} r_l^t \cdot d_{ln}, \tag{2}$$

where $\mathcal{N}_n^t$ is the collection of nodes accessing item t from its replication point at node n, r_l^t is the request rate for topic t from node l and d_{ln} is the distance (in hops) from node l to node n.

And for the overall network traffic from Eqs. (1) and (2), we get:

$$C = \sum_{t=1}^{T} C_t = \sum_{t=1}^{T} \sum_{n=1}^{k_t} \sum_{l \in \mathcal{N}_n^t} r_l^t \cdot d_{ln}. \tag{3}$$

The minimization of the overall traffic cost is given by the minimization of the following constrained nonlinear multivariable function.

$$\min\ (C(k_1, k_2, \ldots, k_t)) \text{ such that, } \begin{cases} \sum_{t=1}^{T} k_t \leqslant L \cdot M, \\ 1 \leqslant k_t \leqslant M,\ \forall t \in T. \end{cases} \tag{4}$$

5.5. Alternatives on the assignment phase

In this section we describe an alternative assignment mechanism which is similar to the Weighted Round Robin

(WRR) scheduling discipline. In Section 5.3 topics were assigned sequentially based on their weights. Particularly, the topic with the largest weight was assigned first, then the topic with the second largest weight and so on. In the WRR alternative the sequence of the assignment does not change but the number of storing points that each topic is assigned to is based on its relative weight. The relative weight ξ_t of topic t is $\xi_t = \left\lceil \frac{\sum_{i=1}^{N} r_i^t}{\min_{\{t \in T\}} \left\{ \sum_{i=1}^{N} r_i^t \right\}} \right\rceil$. This means that the ξ of the less weighted topic is equal to one. The ξ of all topics generate an integer vector of the form $[\xi_1, \xi_2, \ldots, \xi_t]$ where ξ_1 is the relative weight of topic 1 and so on (e.g. [3 1 2 2] means that out of the four topics, topic 2 is the the one with the smallest weight while topics 3 and 4 are twice as large as topic 2 and topic 1 is the largest and its weight is three times larger than topic 2). The WRR alternative of the assignment procedure assigns at each round r

$$k_t^r = \min \left\{ \xi_t \cdot k_t, k_t - \sum_{r'=0}^{r-1} k_t^{r'} \right\},$$

where

$$k_t^0 = 0, \quad \forall t \in T$$

stores to each topic until all topics are assigned to k_t different stores.

In the example of Fig. 3, the vector of the relative weight of the topics is [3 5 1]. Of course the assignment procedure of WRR alternative in that example is the same to the assignment procedure described in Section 5.3 since $k_t = k = 2$ but in the case that $k = 6$ then the assignment of WRR would have been:

Round 1: [3 5 1] stores for each topic.
Round 2: [3 1 1] (topic 2 has already been assigned to 5 stores)
Round 3: [0 0 4] (topic 1 and 2 have been assigned to $k = 6$ stores).

The WRR alternative is fairer to the less weighted topics and as shown in the performance evaluation this lead to better performance regarding the clients' perceived delay and the overall network traffic.

6. Performance evaluation

In this section, we evaluate the proposed storing mechanism using a discrete event simulator. The simulator is written in MATLAB based on the event-driven technique for continuous-time modeling. Before implementing the modified greedy algorithm and the two alternative assignment algorithms we made sure that the implemented greedy algorithm is inline with the algorithms presented in [14,15] (the performance of our greedy implementation is inline with the plots of those two greedy implementations). N brokers are organized in a tree topology (common topology in overlay pub/sub networks) and clients dynamically request on each broker i for stored content with rate r_i^t different for each topic t. We assume that in our network exist T topics and based on the set of experiments each topic should be either replicated at least min_{kt} times or a predefined number of M stores should be placed and appropriately assigned to the topics. Also, each store has a capacity of L different topics. For the purpose of this paper, we assume that there are no limits in the workload (in requests/s) that each store can serve. Finally, the assignment of replicas to the topics is based on their actual weight w_t with the constraint that at least min_{kt} replicas should be assigned to each topic t.

It is widely acknowledged that content-centric pub/sub research lacks public data sets for meaningful evaluation. Thus, synthetic workload generation is widely accepted in the field, under the assumption that the workload generated meets a set of realistic assumptions. Each topic is characterized by two parameters: *popularity* and *locality*. Popularity refers to the request rate related to a topic and locality to the region of the topology likely to originate requests. p_t (respectively l_t) denotes the popularity (respectively the locality) associated to a topic t. Popularity and locality values are computed using a Zipf law of different exponents s_p and s_l respectively. Requests are issued from a set of nodes computed using l_t. Particularly $\lceil l_t \cdot N \rceil$ brokers are potential issuers of requests related to topic t. This set of brokers is computed by choosing a random central node and $\lceil l_t \cdot N \rceil - 1$ additional nodes among the closest nodes to the central node (executing a Breadth First Search algorithm).

Having selected the M stores and assigned to them the T topics using our two assignment alternatives ("*p/s_seq*" for the sequential assignment mechanism and "*p/s_wrr*" for the weighted round robin-like assignment mechanism) we let the system operate under the dynamic client environment. We compare it firstly to the case where each topic is assigned to the k_t stores produced by the first Step of the placement algorithm ("*grd_opt*") described in Section 5.3 disregarding of the storage capacity and the total number M of used stores, and secondly to the case where there is no differentiation among topics during the selection of the M stores and the final assignment of the topics to k_t stores is random ("*rnd*"). The metrics we are interested in are:

- The *overall network traffic, ONT* (in $req \cdot hops/sec$) after the completion of the placement/replication algorithm.
- The *mean hop distance* which corresponds to the mean number of hops between a responding store and the client making the request. This metric is indicative of the response latency as a function of hops in the network.

The above metrics are random variables and we estimate their mean by simulating thousands of observations. We made two sets of experiments; one evaluating both the planning and the assignment phases (see Section 5.4) of the proposed algorithm and one evaluating only the reassignment of topics after an initial planning.

6.1. Overall evaluation of the placement and replica assignment algorithm

In the first set of experiments we conducted two subsets of experiments one assuming a predefined minimum

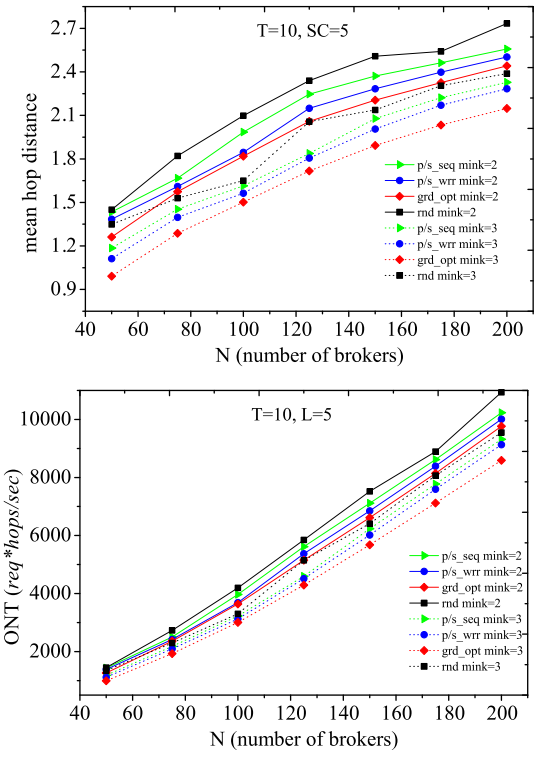

Fig. 4. Performance of the proposed placement algorithm (both assignment alternatives "seq" and "wrr") compared to the "grd_opt" and the "rnd" vs. the number of the brokers in the network.

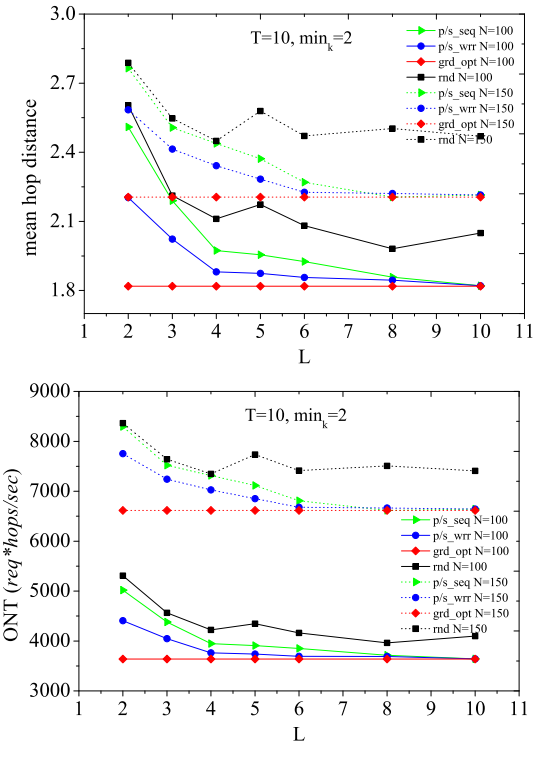

Fig. 5. Performance of the proposed placement algorithm (both assignment alternatives "seq" and "wrr") compared to the "grd_opt" and the "rnd" vs. the storage capacity of the stores in the network.

replication degree for each topic and one assuming a predefined number of stores that should be installed in the network.

6.1.1. Predefined minimum replication degree

In this subset of experiments, we assumed that the Zipf's exponent value of the popularity is $s_p = 1$ while we assumed uniform locality among the topics. Uniform locality implies that requests are generated from every node in the network for every topic or else, the neighborhood of interest for each topic is the whole network. We also assumed that $min_{kt} = min_k = 2$ (minimum replication degree). We mentioned above that the assignment is weighted based on the w_t of each topic, meaning that the number of replicas of each topic is given by $k_t = \left\lceil \frac{w_t}{w_{t'}} \cdot min_k \right\rceil$ where $t' \in T$ is the less weighted topic. Also in our network exist $T = 10$ different topics and the clients' request rate per topic t is $25 \cdot p_t$ requests/s from each broker of the network. We particularly made three different experiments, one varying the number of brokers in the network, one varying the storage capacity of each potential store and one varying the minimum replication degree min_k of the topics in the network.

Figs. 4–6 show the mean hop distance and the overall network traffic for each one of the three different experiments. The proposed algorithm behaves better than the "rnd" algorithm (5%–25% better performance) and close to the "grd_opt" (less than 10% worse), which does not have any constraints regarding the storage capacity and the total number of installed stores. This performance is achieved regardless of the size of the network, the capacity of the stores or the minimum number of replicas installed for each topic. The mean hop distance and the ONT increase sublinearly with the size of the network (Fig. 4) while increasing the L of every store the two proposed alternatives and the "rnd" algorithms install more topics in "privileged" brokers leading to smaller response delays (and loading with less traffic the network) for every request (Fig. 5). Moreover, both the mean hop distance and the ONT decrease as the minimum replication degree (min_k) for each topic increases, since now requests reach closer stores (Fig. 6).

The "wrr" assignment alternative behaves better than the "seq", (4%–17% better performance in every conducted experiment), since as explained in Section 5.5 this alternative results in a fairer assignment of the topics. This means that less popular topics still have the chance to select stores that match their choices (Step 1 of the proposed algorithm) leading to better performance for the whole network. As observed by Figs. 5 and 6, the mean hop distance and the ONT graphs have the same form since the storage capacity of each store and the minimum replication degree of each topic does not alter the overall amount of traffic (req/s) generated in the network, and the ONT is the product of the distance (from a client to the closest store) and the amount of requests generated by the clients of the network. Of course, the addition of new brokers (and new clients attached to them) increases the amount of traffic in the network and

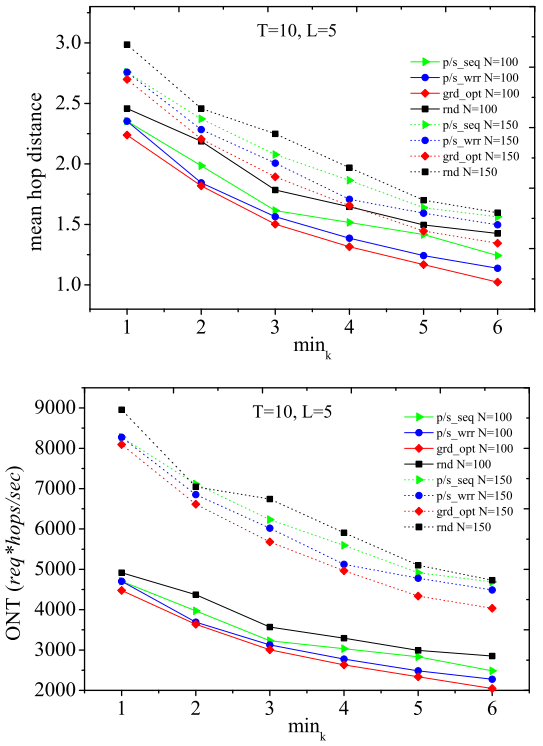

Fig. 6. Performance of the proposed placement algorithm (both assignment alternatives "seq" and "wrr") compared to the "grd_opt" and the "rnd" vs. the minimum replication degree of the topics in the network.

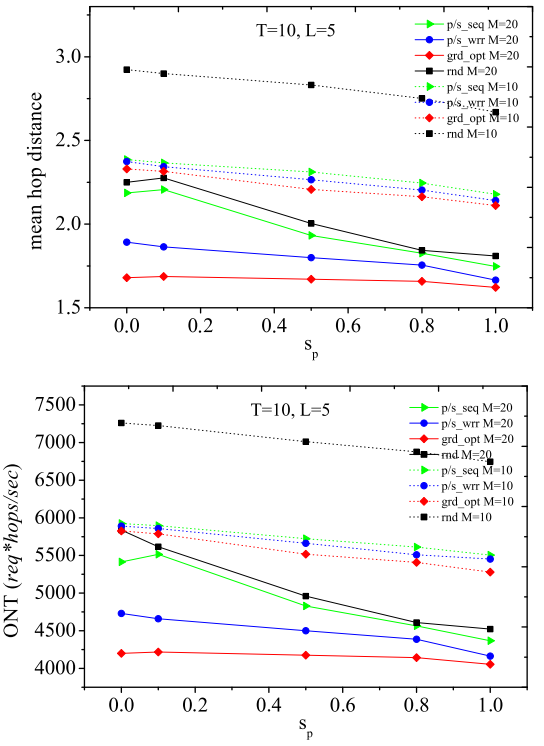

Fig. 7. Performance of the proposed placement algorithm (both assignment alternatives "seq" and "wrr") compared to the "grd_opt" and the "rnd" vs. the exponent value s_p of the popularity.

the mean distance between clients and stores; that's why the ONT graph in Fig. 4 behaves differently from the mean hop distance graph.

6.1.2. Predefined total number of stores

In this subset, we set two different experiments, one assuming uniform locality and vary the exponent s_p of the popularity and one assuming uniform popularity ($s_p = 0, p_t = p = 0.1$ when $T = 10$; uniform popularity means same request rate for each topic equal to 2.5 req/s) and vary the exponent s_l of the locality. Moreover we assumed that there are $M = 20$ and $M = 10$ available stores ($L = 5$ for each store) that should be placed and assigned (based on the weights) to the $T = 10$ topics when the network is composed by $N = 100$ nodes.

Figs. 7-8 show the mean hop distance and the overall network traffic for each one of the two experiments. As previously, the proposed algorithm behaves better than the "rnd" algorithm and close to the "grd_opt" when the popularity exponent changes. Particularly the proposed algorithm performs 10%–25% better than the random algorithm and less than 9% worse than the greedy optimal algorithm, which has no limitations in the number of installed stores and their storage constraints. On the other hand, in the experiment where we change the locality exponent, the proposed algorithm performs four times worse than the "grd_opt" but requires three times less stores. So when the offered stores are predefined, and the "grd_opt" cannot be used, the proposed algorithm (both the assignment alternatives) performs significantly well.

As previously, the mean hop distance and the ONT graphs have the same form since both the popularity and the locality exponent do not alter the overall amount of traffic (req/s) generated in the network but only the way that this amount of traffic is allocated among the topics and the nodes of the network. For that reason the ONT graph is not depicted in the following experiments.

We have also conducted an experiment (Fig. 9; each point is the average of twenty different runs/different combinations of storage failures) assuming storage failures when locality is uniform and the the exponent s_p of the popularity is $s_p = 0.8$. Despite the fact that the proposed planning algorithm and the two assignment alternatives are not designed to take into consideration the possibility of failure of each store during the selection of the stores, we observe a linear increase in the mean hop distance when storage failures occur. Particularly we observe up to 11% increase in the mean hop distance when 25% of the stores fail (fail to serve requests). In the case of a failure of a store a request will be served from the nearest online store, so the observed increase in the mean hop distance is the distance between the new serving store and the one that failed.

6.2. Evaluation of the reassignment phase

In this set of experiments, we evaluate only the reassignment phase of the proposed algorithm (Step 4 in Section 5.3) after an initial planning and assignment of

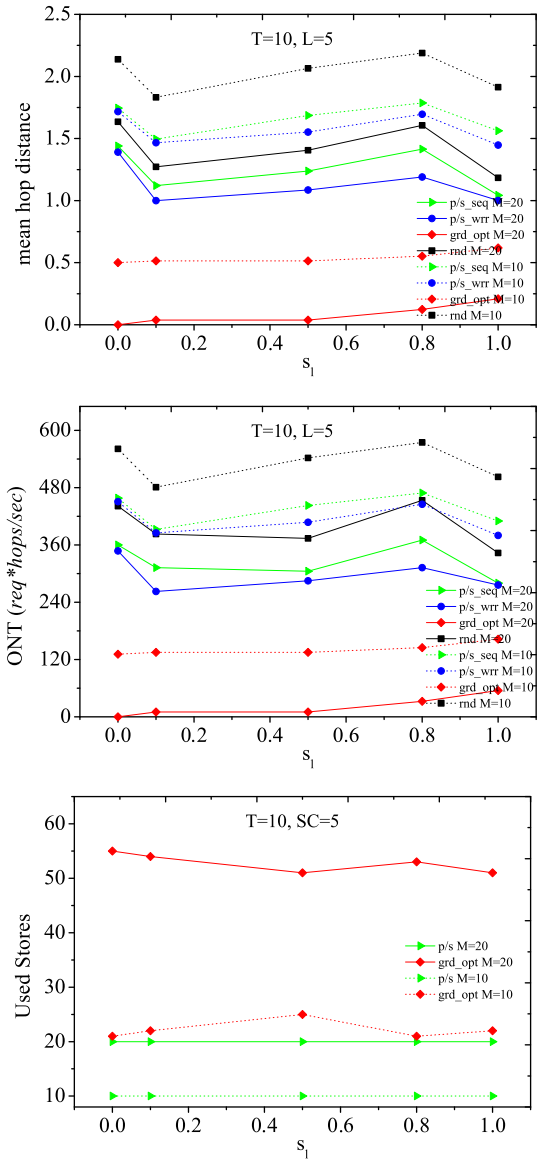

Fig. 8. Performance and total number of installed stores in the network of the proposed placement algorithm (both assignment alternatives "seq" and "wrr") compared to the "grd_opt" and the "rnd" vs. the exponent value s_l of the locality.

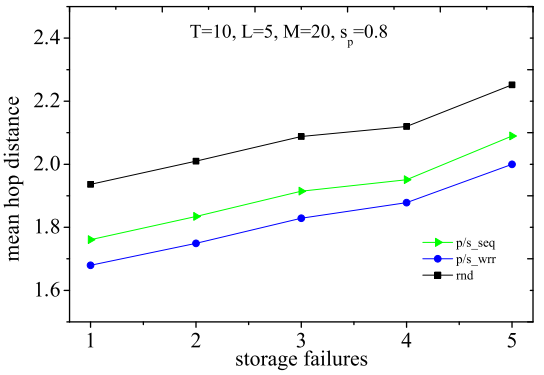

Fig. 9. Performance of the proposed placement algorithm (both assignment alternatives "seq" and "wrr") compared to the "rnd" vs. the number of storage failures.

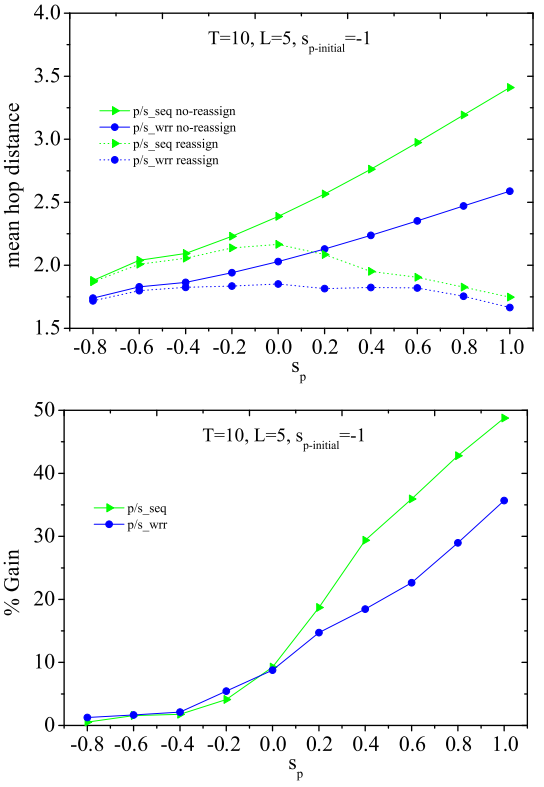

Fig. 10. Performance and % gain of the assignment phase (both alternatives "seq" and "wrr") of the placement algorithm after an initial planning compared to the placement algorithm without reassignment vs. the evolution of the value of the popularity exponent.

the stores. Particularly, we made two different experiments. In the first one, we assumed uniform locality and vary the popularity when the initial planning was done assuming $s_p = -1$ (the last topic is the most popular). In the second experiment we assumed uniform popularity and vary the locality when the initial planning was done assuming $s_l = -1$ (the last topic was requested from the largest neighborhood). Also there are $M = 20$ available stores ($L = 5$ for each store) that should be placed and assigned (based on the weights) to the $T = 10$ topics while the network is composed by $N = 100$ nodes.

Figs. 10 and 11 present the mean hop distance and the relative gain of the reassignment process for the two proposed assignment alternatives. It is obvious that the reassignment of topics manages to retain the good performance of the network even if the popularity or the locality pattern changes radically. Particularly, when the patterns of the popularity and the locality are inverted the reassignment phase by itself delivers up to a 55% decrease in the mean hop distance compared to the case where the assignment of the topics among the stores do not change after the initial planning. Moreover the perfor-

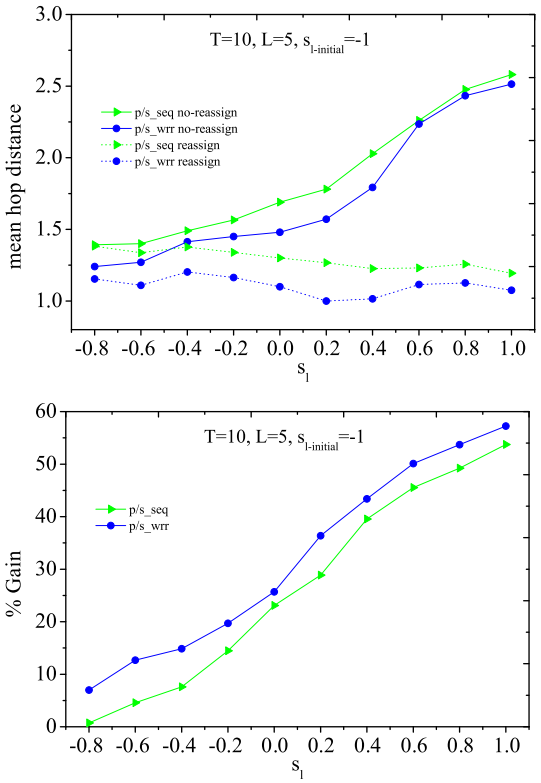

Fig. 11. Performance and % gain of the assignment phase (both alternatives "seq" and "wrr") of the placement algorithm after an initial planning compared to the placement algorithm without reassignment vs. the evolution of the value of the locality exponent.

mance of the reassignment phase is less than 8% worse compared to the performance of executing both the planning and the assignment steps after the change of the popularity and the locality pattern (Figs. 7 and 8).

The relative gain graphs of Figs. 10 and 11 could also be used as a benchmark for the storage provider in his decision to reassign or not the topics in the stores of the network upon the detection of a change in the popularity or the locality pattern. Particularly, when the popularity pattern (the exponent value s_p) changes up to 50% from its initial value the reassignment of the topics has less that 10% impact in the decrease of the mean hop distance and the ONT. This means that a storage provider could skip the reassignment of the topics since the initial planning and assignment still performs quite well. On the other hand, when the locality pattern changes more than 25% from its initial value the reassignment phase is necessary since it can decrease both the mean hop distance and the ONT at least 15%.

6.3. Discussion

From the above performance analysis we observe that the performance of the proposed algorithms (planning and the two assignment algorithms) is at any case up to 25% better than the "rnd" even in the cases where the mean hop distance is less than 3 hops. Of course the proposed algorithms behave worse than the "grd_opt" which has no limitations in the number of the installed stores. Particularly, the proposed algorithms perform very close to the "grd_opt" in the majority of the conducted experiments (1%–15% worse). Only in the case where we change the locality exponent the proposed algorithms behave up to four times worse than the greedy installing on the other hand three times less stores. This implies that in the real world, where a storage provider has limitations in the number of stores that can install, the use of the proposed algorithms is an appropriate solution in almost any scenario.

7. Conclusion and future work

In this paper, we put forward a new mechanism for storing in content-centric pub/sub networks. The proposed concept equips the pub/sub with the ability to store and retrieve stored information. Moreover, we presented a new placement and replica assignment algorithm that differentiates classes of content. Evaluation via simulations of the performance of the system regarding the clients' response latency and the overall network traffic shows that our placement and replica assignment algorithm is a promising solution in almost any scenario. Finally, the two proposed assignment alternatives could also be used regardless of the initial planning of the stores to retain good performance of the network when both the popularity and the locality of the requests change. This work can be extended in many ways such as optimizing different objectives to serve different QoS metrics and SLAs among the storage providers and the content providers.

References

[1] M.K. Aguilera, R.E. Strom, D.C. Sturman, M. Astley, T.D. Chandra, Matching events in a content-based subscription system, in: Proceedings of 18th ACM PODC Atlanta, May, 1999.
[2] A. Carzaniga, D. Rosenblum, A. Wolf, Design and evaluation of a wide-area event notification service, ACM Transactions On Computer Systems 19 (2001) 332–383.
[3] B. Segall, D. Arnold, Elvin has left the building: A publish/subscribe notification service with quenching, in: Proceedings of AUUG, Brisbane, Australia, September 3–5, 1997, pp. 243–255.
[4] G. Cugola, G. Picco, REDS, A Reconfigurable Dispatching System, in: Proceedings of 6th International workshop on Software Engineering and Middleware, Oregon, 2006, pp. 9–16.
[5] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N. Briggs, R. Braynard, Networking named content, in: Proceedings of the 5th ACM CoNEXT, Rome, Italy, December 1–4, 2009.
[6] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, I. Stoica, A Data-Oriented (and Beyond) Network Architecture, in: Proceedings of SIGCOMM, 2007.
[7] PURSUIT FP7 EU project, http://www.fp7-pursuit.eu/PursuitWeb/.
[8] SAIL FP7 EU project, http://www.sail-project.eu/.
[9] 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: Proceedings of DEBS, Toronto, Canada, 2007, pp. 80–84.
[10] V. Sourlas, G.S. Paschos, P. Flegkas, L. Tassiulas, Caching in content-based publish/subscribe systems, in: Proceedings of IEEE Globecom, Honolulu, USA, December 2009.
[11] V. Sourlas, P. Flegkas, G.S. Paschos, D. Katsaros, L. Tassiulas, Storing and Replication in Topic-Based Publish/Subscribe Networks, in: Proceedings of IEEE Globecom, Miami, USA, December 2010.
[12] B. Li, M.J. Golin, G.F. Ialiano, X. Deng, On the Optimal Placement of Web Proxies in the Internet, in: Proceedings of INFOCOM, March, 1999.

[13] I. Cidon, S. Kutten, R. Soffer, Optimal allocation of electronic content, in: Proceedings of INFOCOM, Anchorage, April 2001.
[14] J. Kangasharju, J. Roberts, K. Ross, Object replication strategies in content distribution networks, Computer Communications 25 (2002) 376–383.
[15] L. Qiu, V.N. Padmanabhan, G. Voelker, On the placement of web server replicas, in: Proceedings of IEEE INFOCOM, Anchorage, USA, April 2001.
[16] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit, Local search heuristics for k-median and facility location problems, in: Proceedings of 33rd ACM Symposium on Theory of Computing, 2001.
[17] M. Charikar, S. Guha, Improved combinatorial algorithms for facility location and k-median problems, in: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, October 1999, pp. 378–388.
[18] M. Charikar, S. Khuller, D. Mount, G. Narasimhan, Facility location with outliers, in: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Washington DC, January 2001.
[19] D.B. Shmoys, E. Tardos, K.I. Aardal, Approximation algorithms for facility location problems, in: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 265–274.
[20] E. Cronin, S. Jamin, C. Jin, T. Kurc, D. Raz and Y. Shavitt, Constrained mirror placement on the Internet, in IEEE JSAC, 36(2), September 2002.
[21] M. Karlsson, Ch. Karamanolis, M. Mahalingam, A Framework for Evaluating Replica Placement Algorithms, 2002 (<http://www.hpl.hp.com/techreports/2002/HPL-2002-21>).
[22] R. Cohen, L. Katzir, D. Raz, An efficient approximation for the generalized assignment problem, Information Processing Letters 100 (2006) 162–166.

Vasilis Sourlas was born in Athens, Greece, in 1980. He received his Diploma degree from the Computer Engineering and Informatics Department, University of Patras, Greece, in 2004 and the M.Sc. degree in Computer Science and Engineering of the Computer Engineering and Informatics Department, University of Patras, Greece in 2006. Since 2007 he is a PhD student in the Department of Computer and Communication Engineering, University of Thessaly (Volos), Greece.

Paris Flegkas is currently an adjunct lecturer and a post-doctoral researcher at the Department of Computer Engineering and Telecommunications, University of Thessaly, Greece. He received a Diploma in Electrical and Computer Engineering from the Aristotle University, Thessaloniki, Greece, an MSc with distinction in Telematics (Communications & Software) and a PhD from the University of Surrey, UK, in 1998, 1999 and 2005 respectively. He has been a Research Fellow working in EU and UK national projects for more that 4 years in the Centre for Communications Systems Research (CCSR), UK and 4 years working on EU and Greek projects in CERTH. His research interests are in the areas of policy-based networking, management technologies, traffic engineering, service management, IP QoS and publish/subscribe networks.

Georgios Paschos received his diploma in Electrical and Computer Engineering (2002) from Aristotle University of Thessaloniki, and his Ph.D. degree in Wireless Networks (2006) from ECE dept. University of Patras, both in Greece. Dr. Paschos spent one year in VTT, Finland, working in the team of prof. Norros under an ERCIM postdoc fellowship. Since June 2008, he is working as a research associate for "The Center of Research and Technology Hellas" CERTH, Greece, cooperating with prof. Tassiulas. His main interests are in the area of wireless communications and stochastic modeling.

Dimitrios Katsaros was born in Thetidio (Farsala), Greece in 1974. He received a B.Sc. in Computer Science from 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 with 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" (September–October 2009), and translator for the greek language of the book "Google's PageRank and Beyond: The Science of Search Engine Rankings". His research interests lie are in the area of distributed systems, and in particular, on the Web/Internet, mobile and pervasive computing, mobile/vehicular ad hoc networks, wireless sensor networks.

Leandros Tassiulas obtained the Diploma in Electrical Engineering from the Aristotelian University of Thessaloniki, Greece in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland, College Park in 1989 and 1991 respectively. He is Professor in the Dept. of Computer and Telecommunications Engineering, University of Thessaly, since 2002. He has held positions as Assistant Professor at Polytechnic University New York (1991–95), Assistant and Associate Professor University of Maryland College Park (1995–2001) and Professor University of Ioannina Greece (1999–2001). His research interests are in the field of computer and communication networks with emphasis on fundamental mathematical models, architectures and protocols of wireless systems, sensor networks, high-speed Iinternet and satellite communications. Dr. Tassiulas is a Fellow of IEEE. He received a National Science Foundation (NSF) Research Initiation Award in 1992, an NSF CAREER Award in 1995, an Office of Naval Research, Young Investigator Award in 1997 and a Bodosaki Foundation award in 1999. He also received the INFOCOM 1994 best paper award and the INFOCOM 2007 achievement award.