Storage Planning and Replication Assignment in Content-Centric Publish/Subscribe Networks

Vasilis Sourlas[a,b], 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).*

Abstract

Content-centric publish/subscribe networking is a flexible communication model that meets the requirements of the content distribution needs in the Internet, where the proliferation of content distribution requires information 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 joins 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 replication algorithm which differentiates classes of content and minimize the clients response latency and the overall traffic of the network. Moreover we present and compare two replication assignment alternatives and examine their performance when both the locality and quantity of users request change. The performance of our proposed placement and replication 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 context and performs less than

[☆] V. Sourlas' work was supported by the Greek Ministry of National Education and Religious Affairs (E.S.P.A.- "HRAKLEITOS II"). This work has also been supported by the European Commission through the FP7 PURSUIT program, under contract ICT-2010-257217.

Email addresses: vsourlas@inf.uth.gr (Vasilis Sourlas), pflegkas@inf.uth.gr (Paris Flegkas), gpasxos@inf.uth.gr (Georgios S. Paschos), dkatsar@inf.uth.gr (Dimitrios Katsaros), ltassiuals@inf.uth.gr (Leandros Tassiulas)

1%-15% (regarding the conducted experiment) worse than a greedy (near optimal) approach installing up to 3 times less storages in the network and providing the necessary differentiation among the classes of the content.

Keywords: storage planning, replication placement, content-centric publish/subscribe networks.

---

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 dispatcher. Clients act either as publishers, publishing new events in the network, or subscribers by subscribing to the classes of events they are interested in. The event dispatcher (or rendezvous point or event broker or simply broker) on the other hand, is responsible to collect subscriptions and forward 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.

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. However, in a dynamic distributed environment, clients join and leave the network during time, and it is possible that a subscriber joins the network after the publishing 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 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 "storages" in this work) replicate the whole content of a given server and are targeting to speed 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 storages (e.g. the closest one or the one satisfying other criteria such as the load of the candidate storage). Since storages 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 storage only if that storage 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 storages can advertise the class of the content that they have stored so that clients can retrieve that stored content.

- Propose a new algorithm for the selection of M storage points among the N brokers ($M < N$) 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 ($1 \leq min_{kt} \leq k_t \leq max_{kt} \leq M$) of the topic t among the storages) and c) the storage capacity SC of each storage.

- Evaluate through simulations our design of the storing technique and the new placement and replication assignment algorithm.

The objective function 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 storages in the network and given that storage servers have storage limitations.

The rest of the paper is organized as follows. In Section 2 a brief introduction of storing in pub/sub architectures is given, followed by a brief 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 storage location and the replication 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, 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], PSIRP [7], PURSUIT [8] and 4WARD

[9] 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 semantic of the subscription language. Among the pub/sub models the most known are the *topic-based* pub/sub, which enables information consumers to register 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. As mentioned earlier we adopt a hybrid pub/sub scheme where published messages are classified according to their topic while the matching of subscriptions with the publications follow the content-based scheme.

Despite the number of the content-oriented research efforts, caching/ storing and replication have not received attention in the literature. Only in [11], a historic data retrieval pub/sub system is proposed, where databases are connected to various brokers, each associated with a topic to store. In [11] 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 [12] we introduced a new 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 publish subscribe networks is presented in [13].

On the other hand the placement problem, in the context of Content Delivery Networks and Web Proxies, is a thoroughly investigated problem. Particularly in [14] and [15] 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 on the other hand they consider the problem of placing replicas for only one *origin server*. The placement problem is in fact an NP-hard problem when striving for optimality [16], but there is a number of studies [17]-[22] 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 function. This work is also known as the *k-median* problem.

More placement algorithms have been proposed in [16]. Particularly authors firstly formulate the problem as a combinatorial optimization problem, show that this problem 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 storages cooperating in making the replication decisions. Finally in [23] authors introduce a framework for evaluating placement algorithms. Firstly, they classify and qualitatively compare placement algorithms using a generic set of primitives that capture several objective

functions and near optimal solutions, while secondly provide estimates for their decision time using an analytic model. 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 constraints (limited storage capacity of the storage servers).

3. Problem Description

We assume a content-centric pub/sub network with arbitrary topology represented by a graph $G = (N, E)$. T different topics should be stored at M storages, where each storage has the capability to store SC topics. Each topic $t \in T$ should be replicated k_t times, where $min_{kt} < k_t < max_{kt}$. Requests for the topics are generated at various nodes and they trigger the transfer of the requested item from a storage to the node where the request was generated. The proposed mechanism is composed by two phases 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 storages, while in the *Assignment* phase assigns each topic t at exactly k_t different storages with the target to minimize the total traffic load in the network.

Generally in a real content delivery pub/sub network the *Planning* of the network changes rarely, since it requires the reallocation of the storages 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 storages when the locality and the quantity of the request patterns change is such a way that the performance of the network is degraded. Of course a reassignment requires the calculation of the new places for each topic and the transfer of topics in those locations, but as shown later in the performance analysis is an efficient way to retain the performance of the system in 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. Alternatively, at each snapshot of the system in the network, there exists the same number of messages for each topic. The SC parameter is a limitation introduced by the storage providers and refers to the maximum storing capability of each storage in the network. On the other hand the $min_{kt} < k_t < max_{kt}$ parameter (replication degree of each topic) is a limitation introduced by the content providers of the network and refers to the minimum and maximum number of replicas that the content provider is willing to pay for. Finally, the M parameter

refers to the number of storages that a storage provider should install in the network to serve the storage demands of each topic t.

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 on the network. 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 sent 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 Figures 1 and 2.

In order to retrieve stored information, we add to the system three additional types of messages (besides the already existing `Publish()` and `Subscribe()`), `Advertise()`, `Request()` and `Response()`. When a new storage "*str1*" is installed at broker 5 (Figure 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 publication occurs (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 storage (the distance attribute is built hop by

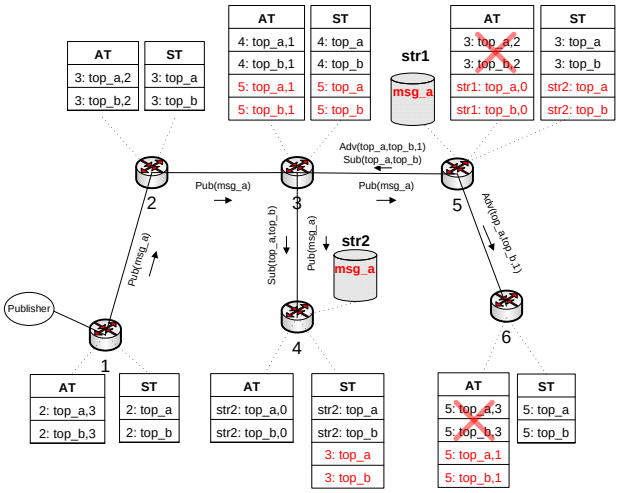

Figure 1: Advertising and Storing of information (in red are the new entries of STs and ATs created by the installation of the "str1"). Publisher at broker 1 publishes a message *msg_a* that matches *top_a* and is stored at "str1" and "str2".

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 `Advertisement Table` (`AT`) which is similar to `ST`. `AT` is a new feature that we added to the pub/sub system. 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 (broker 3 in Figure 1). Keeping more than one entries 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 storage which is closer compared to the other storages already in the `AT`, it adds the advertisement to the `AT`, removes the previous entries and forwards it further (brokers 5 and 6 in Figure 1). Finally, when a broker receives an advertisement for a storage which is further compared to the other storages already in the `AT` simply stops the forwarding of the advertisement without changing the entries of the `AT`.

When a client (client A in Figure 2), is interested in retrieving stored content apart from subscribing (if he/she is also interested for future publications) also makes a request by sending a `Request()` message containing the interested filter ($fltr\_a$). The filter contains the topic that the client

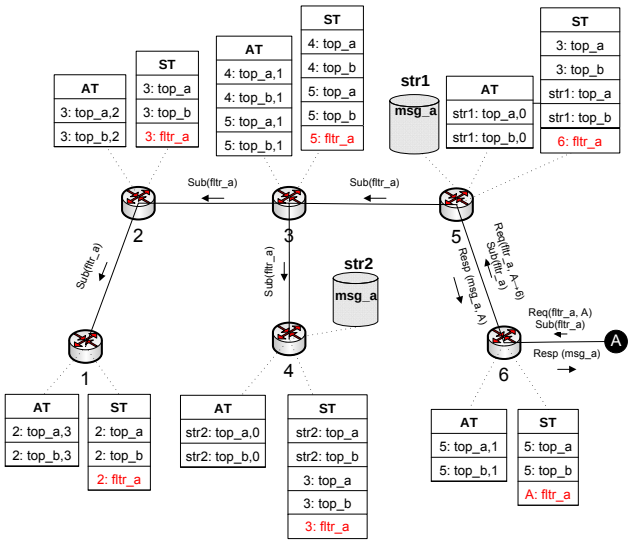

Figure 2: Retrieval of stored information using the request/response mechanism (in red are the new entries of STs created by the subscription of client A).

is interested in. Filters are identical to topics but they can contain more attributes to enable sophisticated (content-based) 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 6 upon receiving the `Request()` message checks in its Advertisement Table (`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 5). Finally, "str1" 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 Figure 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, 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 requesting filter.

5. Placement and Replication Assignment Strategy

We use as the base of our placement and replication assignment scheme algorithms presented in the context of CDN networks. Particularly in [16] and [17], 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 storages based upon both a distance metric and request load, performs the best and very close to the optimal solution.

5.1. Greedy algorithm

Here, we shortly 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/sec) 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 let the propagation delay (hops) from node i to the target server j as D_{ij}. If a storage is placed at node i we define the Gain to be $G_{ij} = P_{ij} \cdot D_{ij}$. This means that the 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:

$$D_{ij} \sum_{l=1}^{N} R_l$$

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 storage at a time (we need k storages 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 storage (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 storage which, in conjunction with the storage already picked, yields the highest Gain. The greedy algorithm iterates until k storages have been chosen for the specific server j.

r_i^t	: request rate for topic t in broker i
N	: number of nodes (brokers) in the network
M	: $(M < N)$ number of storages in the network
k_t	: $(k_t \leq M)$ replication of each topic t in the network
SC	: storage capacity of each storage point in the network
T	: number of classes of content (topics)
w_t	: weight of each topic in the network
SBV	: storage brokers vector
PS_t	: possible stores vector for each topic t

Table 1: Parameters used by the placement/replication algorithm

5.2. Modified greedy algorithm

In the pub/sub network architecture that we assume in this paper, we have no knowledge of the location of the origin server, or differently, there is no such a server. Publishers join the network, publish their content and disappear. So in order to obtain the location of the storages 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 storages. Precisely each vector has N elements with k ones in the index of the selected storages 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 storages are nodes 4 and 6). In two different vectors there might be subsets of possible storages present in both of them. Finally, we select as our storages those k nodes that appeared more times in the per element summation of the N vectors and install at each one a storage following the mechanism described in 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 replication 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 Figure 3 (Table 1 contains all the useful parameters required by the proposed algorithm):

1. For each topic t we execute the modified greedy algorithm presented in Section 5.2 and we get T vector of possible storages PS_t. Regarding the example we get:
   $PS_a = [0\ 3\ 5\ 0\ 2\ 2]$
   $PS_b = [0\ 2\ 5\ 0\ 5\ 0]$
   $PS_c = [0\ 2\ 5\ 0\ 5\ 0]$
   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 (PS_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 regarding the traffic demand of each topic in the network. The weights for the given example are:
   $w_a = 17/50 = 0.34$, $w_b = 27/50 = 0.54$, $w_c = 6/50 = 0.12$
   So the vectors from step 1 are transformed to
   $PS_a = [0\ 1.02\ 1.7\ 0\ 0.68\ 0.68]$
   $PS_b = [0\ 1.08\ 2.7\ 0\ 2.7\ 0]$
   $PS_c = [0\ 0.24\ 0.6\ 0\ 0.6\ 0]$.
3. We select as our storages those M nodes that appeared more times in the per element weighted summation of the T vectors. We call that vector as *storage brokers vector SBV*. 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 $M = 3$ storages in SBV are nodes 3, 5 and 2.
4. For each topic t, starting from the most significant (based on the weight), we assign k_t storages following the procedure below:
   - For each entry in the PS_t of topic t calculated in step 1 assign a storage if that entry also appears in the SBV calculated in step 3 and only if in that storage has been assigned less than SC topics until we get k_t storages (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 $k = 2$ storages. Topic b is assigned to nodes 3 and 5 which were the nodes for topic b appeared more times by 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 storage in SBV with less than $SC = 2$ 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

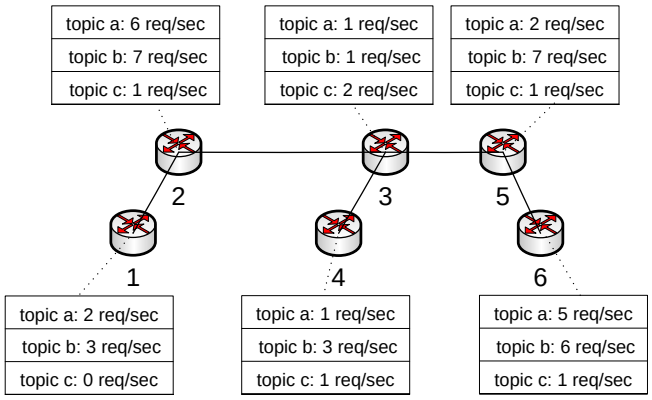

Figure 3: Topology and workload information (requests/second) per each class of content ($T = 3$ topics) together with, $k_t = k = 2$, $SC = 2$ and $M = 3$ form the inputs of the placement algorithm for the pub/sub network.

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 [24].

5.4. Cost Model

Steps 1-3 of the placement 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 of the proposed placement algorithm, which as mentioned above is an NP-complete problem. The access of an information item stored in storage 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 node among those holding the specific item. Thus given that access mechanism we seek to decide the replication 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,

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

where $\mathcal{N}_n^t$ is the collection of nodes accessing item t from its replication points 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 Equations 1 and 2, we get:

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

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

$$\min(C) \text{ such that } \begin{cases} \sum_{t=1}^{T} k_t \leq SC \cdot M \\ min_{kt} \leq k_t \leq max_{kt}, \forall t \in T \end{cases} \tag{4}$$

The above minimization problem is as shown in [16] NP-complete and the Step 4 of the proposed placement algorithm is a heuristic assignment procedure that aims to minimize the overall network traffic cost.

5.5. Alternatives on the placement algorithm

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 serially 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 rw_t of topic t is $rw_t = \left\lceil \frac{\sum_{i=1}^{N} r_i^t}{\min_{\{\forall t \in T\}} \left\{ \sum_{i=1}^{N} r_i^t \right\}} \right\rceil$. This means that the rw of the less weighted topic is equal to one. The rw of all topics generate an integer vector of the form $[rw_1, rw_2, ..., rw_t]$ where rw_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

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

where

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

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

In the example of Figure 3, the vector of the relative weights 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] storages for each topic.
Round 2: [3 1 1] (topic 2 has already been assigned to 5 storages)
Round 3: [0 0 4] (topic 1 and 2 has been assigned to $k = 6$ storages).

The WRR alternative is more fair 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. 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 storages should be placed and appropriately assigned to the topics. Also, each storage has a storage capacity of SC different topics. For the purpose of this paper, we assume that there are no limits in the workload (in requests/second) that each storage 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-based publish/subscribe research lacks public data sets for meaningful evaluation. Thus, synthetic workload generation is widely accepted in the field, under reserve 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 $L_t \cdot N$ brokers are potential issuers of requests related to topic t. This set of brokers is computed by choosing a random central node and $L_t \cdot N - 1$

additional nodes among the most closed nodes to the central node (executing a Breadth First Search algorithm).

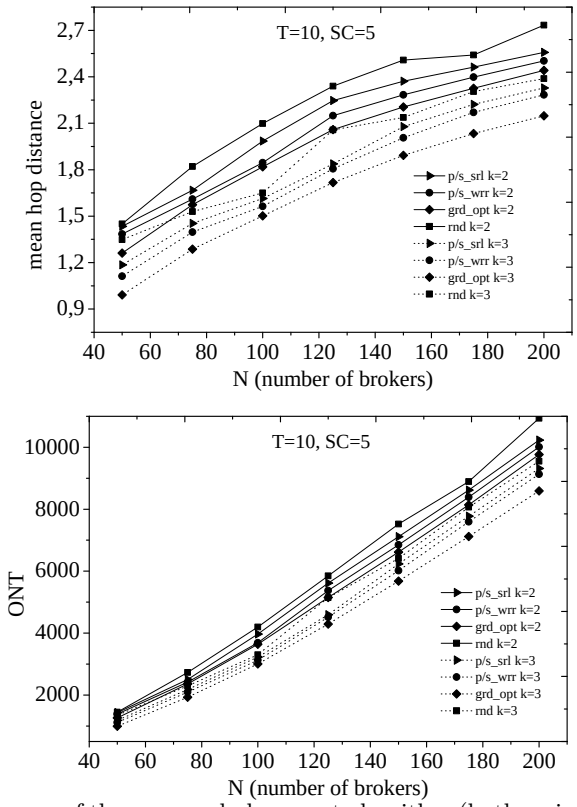

Figure 4: Performance of the proposed placement algorithm (both assignment alternatives "srl" and "wrr") compared to the "grd_opt" and the "rnd" vs. to the number of the brokers in the network.

Having selected the M storages and assigned to them the T topics using our two placement alternative algorithms for pub/sub networks ("*p/s_srl*" for the serial assignment process and "*p/s_wrr*" for the weighted round robin-like assignment process) 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 storages 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 storages, and secondly to the case where there is no differentiation among topics during the selection of the M storages and the final assignment of the topics to k_t storages is random ("*rnd*"). The metrics we are interested in are:

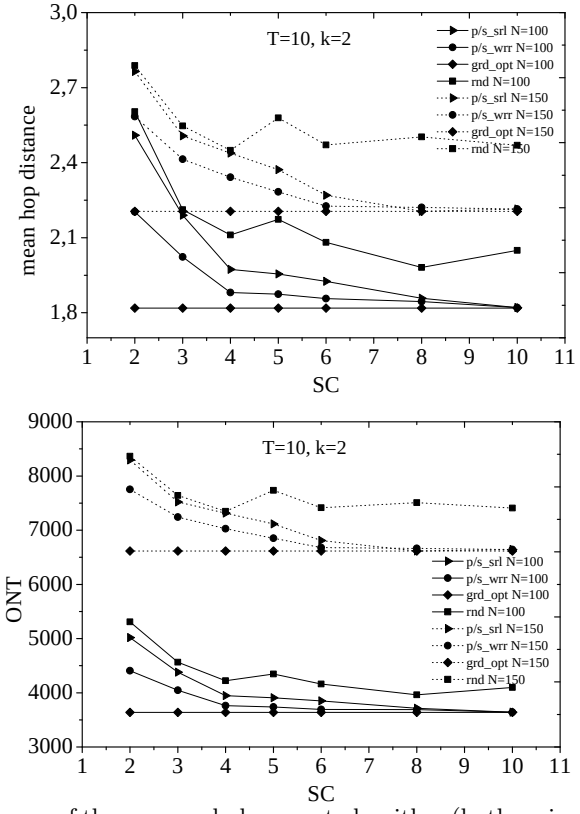

Figure 5: Performance of the proposed placement algorithm (both assignment alternatives "srl" and "wrr") compared to the "grd_opt" and the "rnd" vs. to the storage capacity of the storing points in the network.

- 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 storage and the client making the request. This metric is indicative of the response latency as a function of hops in the network.

We set 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/replication algorithm

In the first set of experiments we conducted two subsets of experiments on assuming a predefined minimum replication degree for each topic and

one assuming a predefined number of storages that should be installed in the network.

6.1.1. Predefined 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 $min_{kt} = min_k = k = 2$. 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 $k_t = \frac{w_t}{w_{t'}} \cdot min_k$ where $t' \in T$ is the less weighted topic. Also in our network exist $T = 10$ different topics and clients request rate per topic t is $25 \cdot P_t$ requests/second from each broker of the network. We particularly set three different experiments, one varying the number of brokers in the network, one varying the storage capacity of each potential storage and one varying the minimum replication degree $min_k = k$ of the topics in the network.

Figures 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 storages. This performance is achieved regardless on the size of the network, the storage capacity of the storages or the minimum number of replicas installed for each topic. The mean hop distance and the ONT increases slower to the increase of the size of the network (Figure 4) while increasing the SC of every storage 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 (Figure 5). Moreover, both the mean hop distance and the ONT are decreasing as the minimum replication degree (k) for each topic increases, since now requests reach closer storages (Figure 6).

The "wrr" assignment alternative behaves better than the "srl", ($4\% - 17\%$ better performance in every contacted experiment), since as explained in Section 5.5 this alternative is more fair is the assignment of the topics. This means that less popular topics still have the chance to select storages that match their choices (Step 1 of the proposed algorithm) leading to better performance the whole network. As observed by the Figures 5-6 the mean hop distance and the ONT graphs have the same form since the storage capacity of each storage point and the minimum replication degree of each

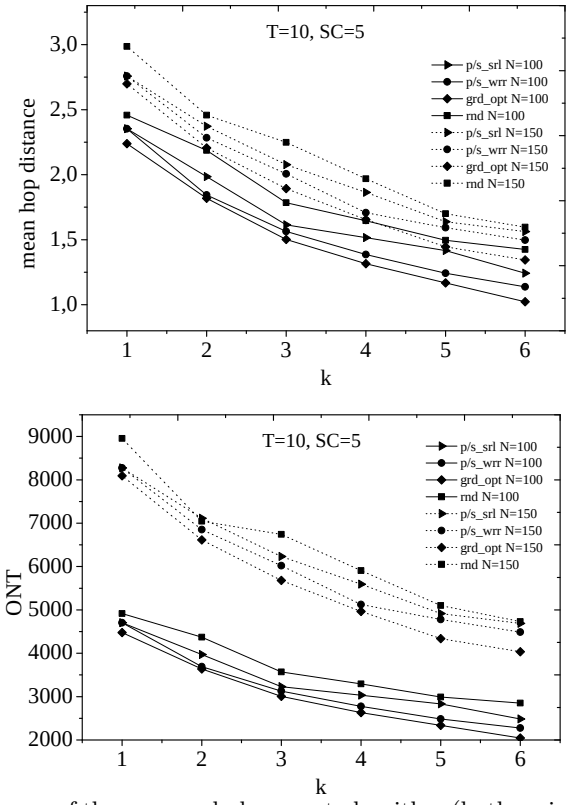

Figure 6: Performance of the proposed placement algorithm (both assignment alternatives "srl" and "wrr") compared to the "grd_opt" and the "rnd" vs. to minimum replication degree of the topics in the network.

topic does not alter the overall amount of traffic (req/sec) generated in the network, and the ONT is the product of the distance (from a client to the closest storage) 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 the mean distance between clients and storages that's why the ONT graph in Figure 4 behaves differently from the mean hop distance graph.

6.1.2. Predefined total number of storages

In this subset, we set two different experiments, one assuming uniform locality and vary the exponential s_p of the popularity and one assuming uniform popularity ($s_p = 0$, $L_p = L = 0.1$ when $T = 10$ uniform popularity means same request rate for each topic equal to 2.5 req/sec) and vary the

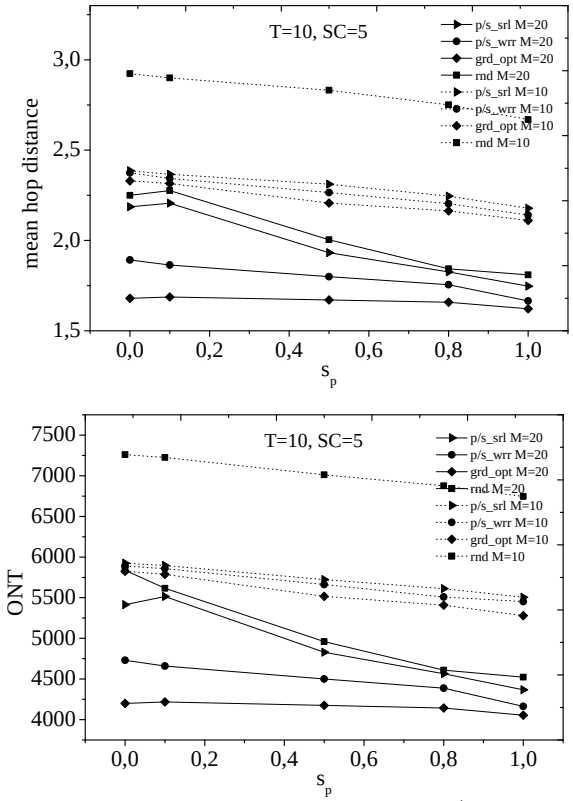

Figure 7: Performance of the proposed placement algorithm (both assignment alternatives "srl" and "wrr") compared to the "grd_opt" and the "rnd" vs. to the exponent value s_p of the popularity.

exponential s_l of the locality. Moreover we assumed that there are $M = 20$ and $M = 10$ available storages ($SC = 5$ for each storage) that should be placed and assigned (based on the weights) to the $T = 10$ topics when the network is composed by $N = 100$ nodes.

Figures 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 from the greedy optimal algorithm, which has no limitations in the number of installed storages and their storage constraints. On the other hand, in the experiment of the changing locality exponent the proposed algorithm performs at least four times worse than the "grd_opt" but requires three times

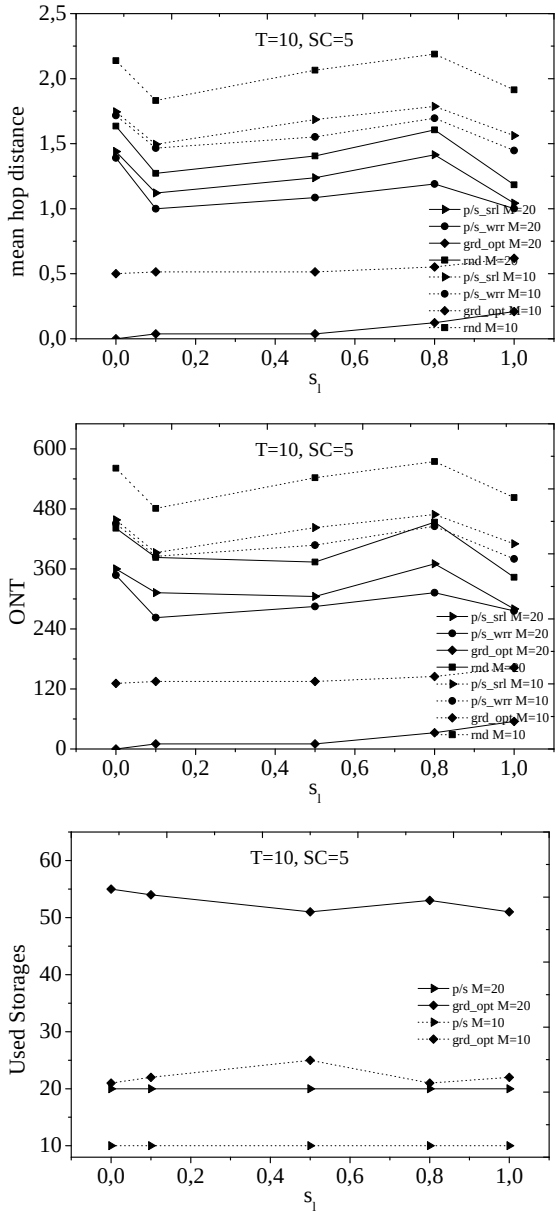

Figure 8: Performance and total number of installed storages in the network of the proposed placement algorithm (both assignment alternatives "srl" and "wrr") compared to the "grd_opt" and the "rnd" vs. to the exponent value s_l of the locality.

less storages. So when the offered storages are predefined, the "grd_opt" cannot be used, the proposed algorithm (both the assignment alternatives)

perform significantly good resulting on average 1.2 - 2.3 mean hop distance on a network of 100 nodes when the offered storages are 20% and 10% of the size of the network respectively and each storage point can store half of the existing topics.

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/sec) 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.

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 of the storages. Particularly we set 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 in 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 storages ($SC = 5$ for each storage) that should be placed and assigned (based on the weights) to the $T = 10$ topics while the network is composed by $N = 100$ nodes.

Figures 9 and 10 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 radically change. Particularly when the patterns of the popularity and the locality are inverted the assignment phase by itself performs up to 55% decrease in the mean hop distance compared to the case where the assignment of the topics among the storages do not change after the initial planning. Moreover the performance 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 (Figures 7 and 8). Thus the performance of both the assignment alternatives makes them an excellent heuristic approach to the minimization of the mean hop distance (and the overall network traffic) presented in Section 5.4.

The relative gain graphs of Figures 9 and 10 could also be used as a benchmark for the storage provider in his decision to reassign or not the topics in the storages of the network upon the detection of a change in

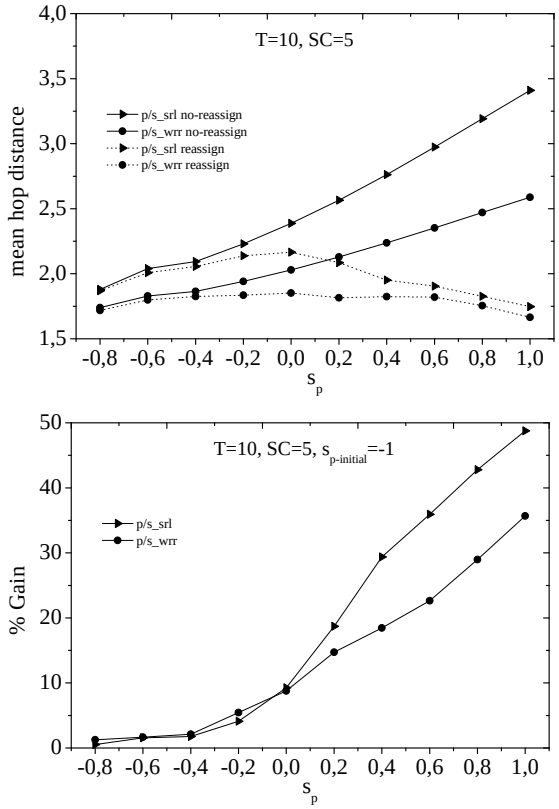

Figure 9: Performance and % gain of the assignment phase (both alternatives "srl" and "wrr") of the placement algorithm after an initial planning compared to the placement algorithm without reassignment vs. to the evolution of the value of the popularity exponent.

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 good. 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%.

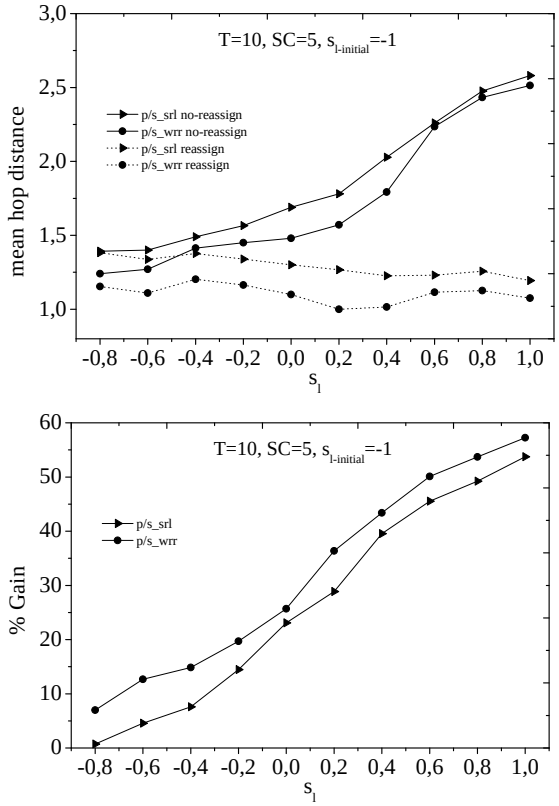

Figure 10: Performance and % gain of the assignment phase (both alternatives "srl" and "wrr") of the placement algorithm after an initial planning compared to the placement algorithm without reassignment vs. to the evolution of the value of the locality exponent.

7. Conclusion And Future Work

In this paper, we put forward a new mechanism for storing in topic-based 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 replication assignment algorithm that differentiates classes of content. Evaluation via simulations that presents the performance of the system regarding the clients response latency and the overall network traffic shows that our placement/replication algorithm is less than 1%-15% worse than the greedy approach installing up to 3 times less storages in the network. Moreover the two proposed assignment algorithms could also be used regardless of the initial planning of the storages to retain the good performance of the network when both the locality and the quantity of the requests change. Particularly the reassignment phase by itself performs less

than 10% worse than re-planning and re-assigning. This work can be extended in many ways such as optimizing different objective functions serving different QoS metrics and SLAs among the storage provider and the content providers.

Acknowledgment

This work is an extended version of the [13] where an initial approach on storing/replication in pub/sub networks is presented.

References

[1] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley and T. D. Chandra, "Matching events in a content-based subscription system," In Proc. of 18th ACM PODC Atlanta, May, 1999.

[2] A. Carzaniga, D. Rosenblum and A. Wolf, "Design and evaluation of a wide-area event notification service," ACM Transaction On Computer Systems, vol. 19, pp. 332–383, 2001.

[3] B. Segall and D. Arnold, "Elvin has left the building: A publish/subscribe notification service with quenching," In Proc. of AUUG, Brisbane, Australia, Sept. 3-5, pp. 243–255, 1997.

[4] G. Cugola and G. Picco, "REDS, A Reconfigurable Dispatching System," In Proc. of 6th Inter. workshop on Software Engineering and Middleware, pp. 9–16, Oregon, 2006.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, R. Braynard, "Networking named content," In. Proc. of the 5th ACM CoNEXT, Rome, Italy, Dec. 1-4, 2009.

[6] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A Data-Oriented (and Beyond) Network Architecture," In Proc. of SIGCOMM, 2007.

[7] M. M Sarela, T. Rinta-aho, and S. Tarkoma, "RTFM: Publish/Subscribe Internetworking Architecture," In ICT-MobileSummit, 2008.

[8] http://www.fp7-pursuit.eu/PursuitWeb/

[9] http://www.4ward-project.eu/index.php?s=file_download&id=39.

[10] Li G., Cheung A., Hou S., Hu S., Muthusamy V., Sherafat R., Wun A., Jacobsen H., and Manovski S., "Historic data access in publish/subscribe," In of Proc. DEBS, pp. 80–84, Toronto, Canada, 2007.

[11] Sourlas V., Paschos G. S., Flegkas P. and Tassiulas L., "Caching in content-based publish/subscribe systems," in Proc of IEEE Globecom, Honolulu, USA, Dec. 2009.

[12] V. Sourlas, P. Flegkas, G. S. Paschos, D. Katsaros and L. Tassiulas, Storing and Replication in Topic-Based Publish/Subscribe Networks to appear in IEEE Globecom, Miami, USA, Dec. 2010.

[13] B. Li, M. J. Golin, G. F. Ialiano and X. Deng, "On the Optimal Placement of Web Proxies in the Internet," In Proc. of INFOCOM, March 1999.

[14] I. Cidon, S. Kutten, R. Soffer, "Optimal allocation of electronic content," In Proc. of INFOCOM, Anchorage, April 2001.

[15] J. Kangasharju, J. Roberts, K. Ross, "Object replication strategies in content distribution networks", Comput. Commun. Elsevier.

[16] L. Qiu, V.N. Padmanabhan and G. Voelker, "On the placement of web server replicas," In Proc. of IEEE INFOCOM, Anchorage, USA, Apr. 2001.

[17] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala and V. Pandit, "Local search heuristics for k-median and facility location problems," In Proc. of 33rd ACM Symp. on Theory of Computing, 2001.

[18] M. Charikar and S. Guha, "Improved combinatorial algorithms for facility location and k-median problems," In Proc. of the 40th Annual IEEE Symp. on Foundations of Computer Science, pp. 378-388, Oct. 1999.

[19] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan, "Facility location with outliers," In Proc. of the 12th Annual ACM-SIAM Symp. on Discrete Algorithms, Washington DC, Jan. 2001.

[20] D.B. Shmoys, E. Tardos and K.I. Aardal, "Approximation algorithms for facility location problems," In Proc. of the 29th Annual ACM Symp. on Theory of Computing, pp. 265-274, 1997.

[21] E. Cronin, S. Jamin, C. Jin, T. Kurc, D. Raz and Y. Shavitt, "Constrained mirror placement on the Internet," in IEEE JSAC, 36(2), Sept. 2002.

[22] M. Karlsson, Ch. Karamanolis and M. Mahalingam, "A Framework for Evaluating Replica Placement Algorithms", http://www.hpl.hp.com/techreports/2002/HPL-2002-21, 2002.

[23] R. Cohen, L. Katzir and D. Raz, "An Efficient Approximation for the Generalized Assignment Problem," Information Processing Letters, Vol. 100, pp. 162-166, Nov. 2006.