2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies

Next Generation CDN services for Community Networks

Vittorio Manetti, Roberto Canonico,
Walter de Donato, Giorgio Ventre
Dipartimento di Informatica e Sistemistica
Universitá di Napoli Federico II
via Claudio 21, 80125 Napoli, ITALY
Email: {vittorio.manetti, roberto.canonico}@unina.it,
{walter.dedonato, giorgio.ventre}@unina.it

Andreas Mauthe, Gareth Tyson
Computing Department, InfoLab21
Lancaster University, Lancaster, UK
Email: a.mauthe@lancaster.ac.uk,
g.tyson@comp.lancs.ac.uk

***Abstract*—Technological developments of the last few years have favoured the creation of distributed networking infrastructures (usually referred to as *Community Networks*) where the resources are made available to the members of a community of people. With emerging large-scale community infrastructures the opportunities for new commercial services and for innovative business models are becoming feasible. Since one of the most resource-demanding services today is access to user-generated content through the web, suitable content delivery services are needed in the context of Community Networks in orders to make an effective usage of shared resources.**

In this paper we describe two architectures we have designed to provide optimized delivery of multimedia web content (e.g. video but also other kinds of User Generated Content) within Community Networks. Both architectures work without any kind of cooperation from the original content providers but assume that a basic web caching service is provided within the network, either by commercial Service Providers or by the community members themselves.

While the first architecture relies on a traditional centralized control entity, the second is designed according to the peer-to-peer paradigm, in order to provide better scalability, robustness to failures and self-configurability. Experimental results are also presented aimed at evaluating the performance gains for end-users from a localized distribution of content in scenarios in which community members are distributed in clusters sparse at different geographic locations.

I. Introduction

The technological developments of the last few years have enabled new forms of interactions and collaboration between individuals, allowing collaboration of sparse communities of people (a.k.a. *Network Communities*) sharing common interests as well as producing synergies in pursuing common objectives. At the application level, these trends have been facilitated by technologies such as Web2.0, social network platforms, mobile computing, and so on. In the last few years this phenomenon has evolved to a more advanced stage, involving new forms of resource sharing and the creation of distributed networking infrastructures (usually referred to as *Community Networks*) which are made available to the members of the community supported by collaboration [1].

A number of Wireless Community Networks (WCNs) have been established to provide Internet access to community members. These networks have been created either through spontaneous collaboration of people who share their xDSL home connection to the Internet, or through the initiative of local institutions. For example, councils and universities have started to offer wireless access to Internet services to user communities (e.g. students) in limited areas or public buildings. The most popular *spontaneous* WCN is the one created by the so called *FON community* [2]. FON members (i.e. Foneros) share some of their home xDSL Internet connection and get worldwide free access to the Communitys WiFi Hot Spots. With the availability of such large-scale community infrastructures opportunities for new commercial services and for innovative business models is becoming reality. The FON community, for instance, has established a business consisting of selling Internet access to those who decide not to share any connection with the rest of the community.

One of the challenges for Community Networks is to provide them with autonomic capabilities for optimal utilization of shared resources. Shared resources in a Community Network may be, for instance:

- bandwidth of Internet connections (e.g. xDSL lines);
- bandwidth of wireless links (e.g. access points, Wireless Mesh routers, ...);
- storage space;
- computational power.

One of the most resource-demanding services today is access to user-generated content through the web. This is just the most basic service to be provided in such kinds of infrastructures, acting as the basis for more complex 'Content Services', enabling users not only to consume content, but also to search for particular pieces of content, to combine audiovisual rendering effects, and to edit complex multimedia objects. Nonetheless, to make an effective usage of the resources made available to the community, proper content delivery services are needed also in the context of Community Networks.

In this paper we describe two architectures that have been designed to provide optimized delivery of web content (such as videos and other kinds of User Generated Content) within Community Networks. Both architectures assume that:

978-0-7695-3786-3/09 $26.00 © 2009 IEEE
DOI 10.1109/NGMAST.2009.45

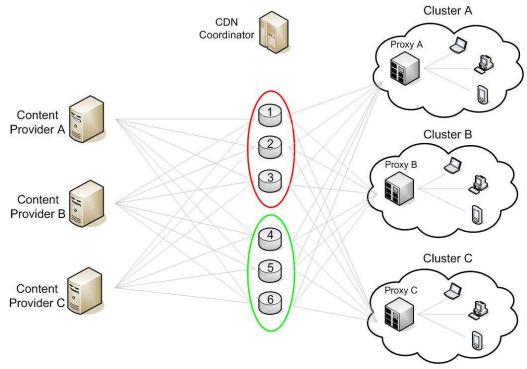

Fig. 1. Centralized Architecture

1) content is delivered through them without any kind of cooperation of the original content providers;
2) a basic web caching service is provided within the network either by commercial Service Providers or by the community members themselves.

While the first architecture relies on a traditional centralized control entity, the second one is designed according to structured peer-to-peer principles, in order to be more scalable, more robust to failures and self-configurable.

We also present initial experimental results showing the performance gain for end-users from localized distribution of content in scenarios in which community members are distributed in clusters, sparsely at different geographic locations.

II. A CDN FOR USER COMMUNITIES WITH CENTRALIZED CONTROL

The goal of this work is to create more flexible and dynamic CDNs that provide on-demand services. Such infrastructures should be able to support a negotiable number of Content Producers and to deliver web content to dynamically configured user communities. To this end, we have defined a novel architecture that manages a set of dynamically located content replicas. Further, there are one or more user communities composed by users that continuously join and leave the network.

The proposed architecture (figure 1) is composed of three different entities: a caching system, *n* Proxy nodes, and a CDN Coordinator.

The caches composing the caching system are provided by third parties (e.g. Internet Service Providers) and are distributed throughout the network. Each cache is used for performing the typical functionalities and, in addition, each of them implements some monitoring and configuration modules.

A Proxy node is assigned for each subset of users (i.e. community). The Web client by which users send content requests, has to be configured in a way to automatically forward such requests to the Proxy node. Knowing the content location scheme and being destination for each content request sent by users, it is redirecting requests to the cache, which offers the desired content. The Proxy implements a content-aware request routing mechanism based on the content location scheme. Processing an incoming request consists of choosing the cache to forward the request to. To this end the Proxy node exploits a Content Routing Table (figure I) composed by entries associating each content with the corresponding cache. In addition to this task, the Proxy node has to be able to forward requests directly to the Original Server when the requested content is not available into the cache system.

Further, the aim is to provide a system that is capable of adapting itself, considering the changing scenarios. In order to achieve this goal, a number of parameters have to be monitored, i.e.: the network condition, the traffic generated by each cluster, the community users' behaviour, and whatever information that could be used to change the caching system configuration and/or the content location scheme. To this end, we introduce the CDN Coordinator, a node that performs centralized control of the entire architecture.

A. Details about the CDN Coordinator functionality

The CDN Coordinator is responsible for the following tasks:

- monitoring the architecture and collecting information;
- computing the optimal content location scheme;
- configuring the caching system;
- updating the Content Routing Table of the Proxy nodes.

Concerning the first task, the CDN Coordinator collects information about: i) the available bandwidth, ii) the traffic generated by each subset of users, iii) the number of requests for each content that is available in the caching system. The CDN Coordinator does not measure these parameters directly, but collects information provided by entities to which this task is assigned. Software (introduced in section II-B) is used to determine the available bandwidth between the network through which the users access to the Internet, and the caching system; the CDN Coordinator periodically acquires this information. Each cache keeps statistical information about each incoming content requests; the CND Coordinator collects such information accessing directly the log files of the nodes composing the caching system.

With this information the CDN Coordinator can perform its main functionality, i.e. computing the optimal content location scheme and reconfiguring the caching system. This work is performed when particular conditions occur. Thus, it can be considered an *event driven* process. After computing the optimal location scheme the CDN Coordinator coordinates all the entities composing the architecture, aiming to impose that the computed request routing policies will be followed.

Starting from the Simple Plant Location classical model, we defined an optimal location scheme that computes an objective function that is based on the following parameters:

- the time needed for the transmission between client and cache, and between server and cache, for each content;
- the cache locating fee;
- the available budget;
- information about the use of the links between each client-cache pair, and each server-cache pair, for each content item;
- the content location scheme;

Prot.	Host	Port	Path	Host	Port
http	www.somehost.com	80	*	Cache1	8080
http	www.somehost.com	80	/path/video.avi	Cache2	3128
http	*	80	/path	Cache3	8080

TABLE I
CONTENT ROUTING TABLE

- the request rate for each content item;
- the number of requests the cache is able to satisfy;
- the size of each content item;
- the total amount of available storage for each cache;

The objective function computed during this process has the goal to minimize the time needed to complete the delivery process for each content item; such function has to be computed taking into account a number of constraints, such as:

- budget constraint: the cost needed to activate the caching system has to be lower than the available budget;
- the clients can request a content item from a specified cache only if the content item was located on that cache;
- each content item located on a cache has to be taken from a single server;
- a client can request a particular content item only on a single cache;
- a content item can be located on a cache only if that cache was activated;
- the total number of requests for a content item on a cache cannot be higher than the number of requests the cache can satisfy;
- the total size of the web objects located in a cache cannot be higher than the total amount of storage available on that cache.

Since the location problem is solved, the CDN Coordinator manages, on one hand, the content uploading process providing instructions to each cache about the content to store and the origin servers to contact. This process is implemented using a push-based approach. It provides, on the other hand, instructions to the Proxy nodes about the computed content location scheme and about the request routing policies that have been selected. Concerning this latter task, using a properly realized communication protocol, the CDN Coordinator directly accesses the Content Routing Table of each Proxy in order to update them.

Each Proxy node manages a Content Routing Table implementing an association between a content and the corresponding cache. Each table entry contains the following information: *key* - *value*, where *key* is the URL of the resource, and *value* is the cache to which the request for this resource has to be redirected. The *Content Routing Management Protocol* (*CRMP*) is the protocol through which the CDN Coordinator accesses to the Content Routing Table of the Proxy nodes in order to insert, update and drop table entries.

B. Details on the CDN Coordinator software architecture

The CDN Coordinator has been implemented as a framework using Java; such framework manages plugins that are used to collect information needed to compute the content location scheme, and a list of Proxies and Caches composing the architecture.

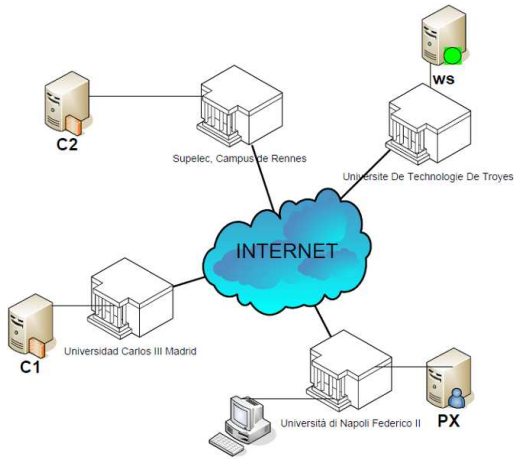

Fig. 2. Testbed realized on Planetlab

Cache nodes have been implemented using Squid, an Open Source proxy cache server; PathChirp is used to measure the available bandwidth between Proxy nodes and caches.

In our prototype we realized two plugins: the *SquidStatsPlugin* and the *PathChirpPlugin*. The first has the task to periodically query the cache nodes to obtain statistical information about the access rate to each content item (this work is performed checking the Squid log files). The PathChirpPlugin acquires information measured by PathChirp about the available bandwidth.

III. PERFORMANCE STUDIES

To evaluate the performance of the proposed architecture in a realistic scenario, we conducted some experiments exploiting the Planetlab [3] infrastructure. We setup a testbed (shown in figure 2) using five nodes of the slice *uninaonelab_cdn*; such nodes are located as follows:

- at University Federico II of Napoli (Italy)
  - Wget client (CL): planetlab01.dis.unina.it
  - Proxy server (PX): planetlab02.dis.unina.it
- at University Carlo III of Madrid (Spain)
  - Cache server Squid (C1): planetlab1.it.uc3m.es
- at University of Technology of Troyes (France)
  - Cache server Squid (C2): planetlab1.utt.fr
- at Supelec center of Rennes (France)
  - web server Apache (WS): pl1.rennes.supelec.fr

The nodes were selected to highlight the advantages obtained using the proposed architecture. In details, the web server is located in a network with a bandwidth limited Internet access and the two caches are geographically far from each other.

The experiments were conducted measuring content delivery time in different scenarios. An object of approximately

	Scenario	$\mu[s]$	$\sigma[s]$
A	CL → C1 → WS	69.9	10.6
B	CL → C1	3.5	0.6
C	CL → PX → C1 → WS	58.7	7.5
D	CL → PX → C1	3.4	0.2
E	CL → C2 → WS	128.6	14.1
F	CL → C2	78.7	13.5
G	CL → PX → C2 → WS	125.5	25.0
H	CL → PX → C2	72.8	7.9

TABLE II
RESPONCE TIME FOR A CONTENT REQUEST

4MB (4.162.048 bytes) was placed on the web server and, for each scenario, eight requests for this content item were executed by the client. The scenarios were set in order to obtain the content:

- directly from the Web Server (CL→WS)
- from the Caches in case of miss (CL→C1→WS and CL→C2→WS)
- from the Caches in case of hit (CL→C1 and CL→C2)
- from the web server through the Proxy (Cl→PX→WS)
- from the Caches through the Proxy in case of miss (CL→PX→C1→WS and CL→PX→C2→WS)
- from the Caches through the Proxy in case of hit (CL→PX→C1 and CL→PX→C2)

We evaluate the response time of the requests, namely the time elapsed between the request and the complete reception of the content. Table II reports the average and the standard deviation of the response time obtained for each scenario.

The results show that obtaining the content from C1 is the best choice, because its bandwidth towards CL is larger than those between CL and WS/C2. Moreover, as C1 is located in Spain and C2 and WS located in France, it demonstrates that a geographical proximity does not imply faster data transfers. Finally, looking at C, E, G and I scenarios results, the impact of the presence of PX between CL and C1/C2 does not degrade the performance. These results confirm that a careful configuration of the content routing table of the Proxy can significantly improve the content delivery time.

IV. A PEER-TO-PEER CDN SERVICE FOR USER COMMUNITIES

In this section the alternative peer-to-peer approach is introduced. The P2P-based architecture is more flexible due to its self-organisation capabilities and thus more suitable for wireless and ad-hoc community content networks.

The *cooperative CDN* has been implemented as a prototype. Its architecture consists of two layers:

- A traditional web caching layer (Amazon S3 [4], for instance,), comprising a set of web caching nodes provided either by third parties (e.g. Internet Service Providers) or by selected community members (e.g. Linux boxes running squid). Such caches are connected to the access infrastructure through broadband symmetric links and they are made available to the community with the specific task of reliably delivering cached content to end-users.
- A structured peer-to-peer system implementing a Distributed Hash Table (DHT) and providing an indexing function. The cooperative peer-to-peer network is formed by end-users that collaborate to redirect users requests towards the most suitable web caching node, with the goal of minimizing the latency perceived during the content delivery process from the selected web cache.

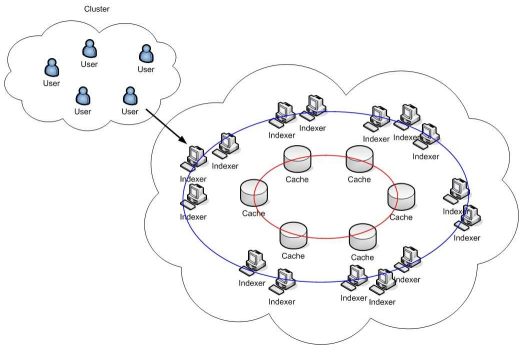

Fig. 3. Architecture

An alternative solution to the use of a traditional caching system, could be the realization of a cooperative peer-to-peer caching mechanism implemented by the users themselves, as well as for the indexing mechanism. Nevertheless, this solution could suffer from churn problems and/or problems concerning the users' connectivity (that is probably asymmetric). We are proposing caching as a service provided by ISPs to communities, rather than a cooperative resource sharing effort from the community itself. ISPs are in a favourable position to place caches in locations with symmetric connectivity , and they probably prefer to deploy traditional web caches in their infrastructures rather than peers of a cooperative web caching p2p system.

A first main difference between the centralized architecture and the P2P-based architecture concerns the lookup mechanism: we are proposing for this latter case the use of a Distributed Hash Table instead of a Content Routing Table replica on each Proxy node. The rationale behind this architectural design choice is that, in order to speed up the deployment of such infrastructures for community networks, we intend to rely on the cooperative effort of community members themselves, and for this reason we did not include in the P2P architecture the complex Content Routing services implemented in the proxies of the centralized architecture.

Likewise, as for the request redirection mechanisms, we assumed that we cannot rely on redirection mechanisms implemented in the original server or in DNS servers, since the CDN is established without support of the content providers. We also assume that ISPs are not willing to do packet introspection to transparently redirect HTTP flows towards our caches. The P2P infrastructure is then established by the community peers to cooperatively implement the redirection schemes, to

discover the web objects in the web caches and to balance the load among the caches.

Finally, as for the management functions, we assume that web objects to be provided by the P2P CDN are still going to be inserted into the DHT by a single user, the CDN administrator, who has the power of creating a reference to a given URL into the DHT.

A. Architectural details

In this subsection we describe in more details *Donizetti*, a hybrid content distribution system that we have designed according to the general principles above presented. The Donizetti architecture is composed of a collection of stable distributed web caches (in the order of hundreds) located as close as possible to a cluster of users and shared among the members of a community. Communities implement a P2P indexing system for locating the cache responsible for each web object. This indexing system is based on, Pastry [5], a scalable and reliable Distributed Hash Table which also provides some form of control over the characteristics of the overlay. Unlike PAST, a distributed file system built on top of Pastry, Donizetti relies on an external web caching system to store the cached web objects, and the DHT is only used to maintain references to the stored objects.

Plain DHTs suffer of the well-known churn problem that occurs when nodes continually join and leave the network in an unpredictable fashion. A classical approach to improve the performance of a DHT under churn is to leverage the natural heterogeneity in the system by using super-peers [6], [7]. Super-peers are selected nodes with extra capabilities, but also extra duties. A super-peer acts as a server to a dynamic subset of weaker (client) peers. These weak peers submit queries to their super-peers and receive results from them. Super-peers are connected to each other forming an overlay network of their own, submitting and answering requests on behalf of the weak peers.

Donizetti refers to a model based on the use of super-peers . The upper layer of the CDN infrastructure consists of a DHT whose peers are located in the user terminal, where they act in cooperation with the web browser. While *normal peers* may be embedded in the browser itself (e.g. as a browser plug-in), super peers need to exist independently of the user browser. For this reason they will be implemented as service daemons running on selected nodes (with the permission of the end-users). Election mechanisms for super peers are still under investigation.

We identify the super-peer nodes as *Indexers*. Each Indexer is linked to a Cache trough a n-to-1 association, and implements a DHT interface trough which it interacts with other Indexers and with normal peers.

Due to the natural organization of users in clusters, we may suppose that all peers in a cluster refer to the same Indexer as first hop for their requests to the CDN.

The we browser of each user terminal is equipped with a plug-in that, considering a specified requested content item, verifies if the content delivery service is available for such content item; if alternative, it sends an HTTP GET directly to the origin web server.

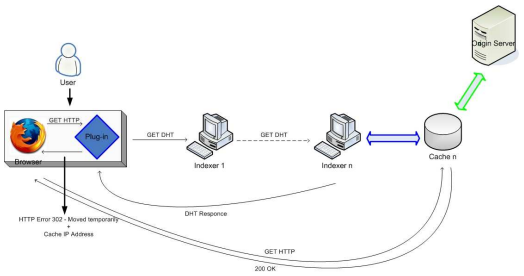

Fig. 4. Content Delivery process

Using its own browser, the user performs a request for a specified content item. Supposing that the content delivery service is available for such content item, the browser sends an HTTP GET request to the local DHT peer, which, in turn, forwards it to the associated super-peer (acting as Indexer). Such request passes through a number of Indexers until it arrives on the node responsible for the requested content; this peer sends a response to the DHT peer running in the user terminal, providing it with the IP address of the Cache to which it is associated. The DHT peer finally redirects the browser to the Cache.

Hence, in Donizetti, DHT nodes do not perform content caching , but they only keep references of the web objects' URLs. This choice has its motivation in the fact that Indexers may be connected to the Internet through an xDSL connection, and not be suitable to serve web objects to the rest of the community.

As for content assignment to caches, we are currently investigating mechanism that allow the system to keep content items as close as possible to users who request them most often. We are also currently defining proper mechanisms to replicate and delete content items throughout the DHT for maintenance and optimisation purposes. We intend to define proper ways to manage, address and discover replicas of the same web object in the caching layer through the p2p indexing layer. One of the strengths of our architecture is that it is self-organizing, i.e. it does not need a central coordination function to manage the allocation of resources.

In order to evaluate the effectiveness of the proposed architecture, we are currently implementing a simulation model of Donizetti, based on the OverSim [8] and HttpTools [9] extensions of the OMNET++ network simulator [10].

V. Related Work

In [11] Cheng et al. present a group-based P2P web caching system named IntraCache. IntraCache looks at how to exploit browser caches of nodes in intranets and use an interest group model to organize peers in the system. Peer interest groups are a set of active peers, who have the same interests and are involved in sharing browser web caches. The proposed overlay network consists of three kinds of peers: fat peers, thin peers and P2P registrar. This latter maintains the peers'

current status information and records the relationship between fat peers and thin peers. Further, the interest group information includes the scale and current interest vector. Fat peers may be a past proxy server or produced dynamically by thin peers in one group. The Peer Manager and Index Builder thread running in the fat peer manages the peer status and index information in one group respectively. In addition, some hot browser cache content items are also store in fat peers. Thin peers are a common node, which can communicate and share resources directly with any node in the P2P network.

In [12] Garbacki et al. present a two-level caching infrastructure; the proposed architecture assumes the existence of caches of two types. Weak peers keep lists of super-peers that proved in the past to be in some way most suitable for them. Super-peers index files residing the weak peers that were recently requested by some peers. The adopted approach consists of placing shared caches at a set of selected (super-) nodes. These caches are used (i.e. shared) by many peers at the same time. A peer that joins the system is automatically associated with one or more super-peers and can immediately use the information collected by these super-peers. The information stored at a node depends on the type of this node. Each weak peer has a super-peer cache which contains the identities of super-peers (e.g., their IP addresses and port numbers). Each super-peer has a file cache of pointers to files stored at some peers. The probability that the search succeeds is high if the requested information is possessed by only one of the super-peers. Whenever a weak peer initiates a search, it first checks the file caches of the super-peers known to it. If the file is not found in one of these caches, a system-wide search in the super-peer network is initiated. The pointer to the located file is then cached by one of the super-peers known to the weak peer that initiated the search.

In [13] Tyson et al. present a peer-to-peer caching architecture called Corelli. This allows users in communities without sufficient resources to build a dedicated caching infrastructure to cooperatively build their own peer-to-peer equivalent. In Corelli communities specific peers monitor the request trends of the network. When these nodes consider the community's behaviour to be cacheable, a selection of the highest capacity peers dynamically instantiate themselves as caching peers. Other peers in the community then forward requests through this virtual cache allowing the caching peers to replicate popular content. As demand varies, the Corelli Cache dynamically expands and contracts to use greater or fewer peer resources to best reflect the requirements of the community. This can occur even to the extent of caches being entirely removed from the community if request trends become uncatchable.

VI. Conclusion and Future Work

Community Networks are shared infrastructures made available to the members of a community of people. The existence of common interests among the community members, as well as the scarcity of communication resources suggest the opportunity of establishing a new kind of Content Deliver Infrastructures particularly suited to this new scenario. In this paper we describe two architectures we have designed to provide optimized delivery of multimedia web content within Community Networks. The first kind of infrastructure is particulary meant as a service provided by a third party, and requires deployment of nodes with different functions within the network. The second one relies on the cooperative effort of community members and is designed according to a p2p model. Both types of infrastructures assign the responibility of content management to a unique administrator user. We have described the main architectural differences of the two architectures and presented some preliminary results of an experimental evaluation conducted on PlanetLab.

VII. Acknowledgements

This work has been supported by the European Union under the IST Content (FP6-2006-IST-507295) project. The CONTENT Network of Excellence targets Content Delivery Networks for Home Users, as an integral part of Networked Audio-Visual Systems and Home Platforms.

References

[1] T. Plagemann, R. Canonico, J. Domingo-Pascual, C. Guerrero, and A. Mauthe, "Infrastructures for Community Networks," *Content Delivery Networks*, p. 367, 2008.

[2] [Online]. Available: http://www.fon.com

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 00–00, July 2003.

[4] [Online]. Available: http://aws.amazon.com/s3/

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, pp. 329–350, 2001.

[6] A. Mizrak, Y. Cheng, V. Kumar, and S. Savage, "Structured superpeers: Leveraging heterogeneity to provide constant-time lookup," in *Proceedings of the 3rd IEEE Workshop on Internet Applications, WIAPP'03*, 2003, pp. 104–111.

[7] Y. Zhu, H. Wang, and Y. Hu, "A super-peer based lookup in structured peer-to-peer systems," in *Proceedings of the 16th International Conference on Parallel and Distributed Computing Systems, PDCS'03*.

[8] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007.

[9] K. Jonsson, "HttpTools: A Toolkit for Simulation of Web Hosts in OMNeT++," in *Proceedings of the 2nd OMNeT++ workshop*, 2009.

[10] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference, ESM01*, 2001, pp. 319–324.

[11] H. Cheng, Z. Gu, and J. Ma, "IntraCache: An Interest group-based P2P Web Caching System," in *Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium, IPDPS'07*, 2007, pp. 1–8.

[12] P. Garbacki, D. Epema, and M. Van Steen, "A two-level semantic caching scheme for super-peer networks," in *Proceedings of the 10th International Workshop on Web Content Caching and Distribution, WCW'05.*, 2005, pp. 47–55.

[13] G. Tyson, A. Mauthe, S. Kaune, M. Mu, and T. Plagemann, "Corelli: A Peer-to-Peer Dynamic Replication Service for Supporting Latency-Dependent Content in Community Networks," in *Proceedings of the 16th Multimedia Computing and Networking Conference, MMCN'09*, 2009.