# A Hierarchical Naming System for Scalable Content Distribution in Large Networks

Yaxiong Zhao†, Jie Wu‡, Cong Liu§, and Mingming Lu¶
†Amazon.com Inc., ‡Temple University, §Sun Yat-Sen Uuniversity, ¶Central South University
Emails: †zhaoyaxi@amazon.com, ‡jiewu@temple.edu, §gzcong@gmail.com, ¶mingminglu@csu.edu.cn

***Abstract*—The Internet has become a platform for delivering and consuming content. Additionally, mobile devices consume a significant chunk of content, which poses a unique challenge for provisioning efficient content delivery and is not addressed in contemporary infrastructures. In this paper, we present a naming system that provides Internet-scale content distribution service with a particular emphasis on content delivery across mobile devices. The basis of the system is a highly flexible and expressive content naming scheme. It is flexible in expressing diverse application requirements. To achieve fast name resolution, we design a two-level indexing network using Distributed Hash Table (DHT). This architecture is open and scalable. It utilizes a key-value storage system built on DHT. Our content delivery protocol relies on *content routers* to balance the load of delivering content and to throttle denial-of-service attacks. The feasibility of the system is analyzed based on a realistic modeling of the Internet. Simulation studies are performed to verify its usefulness and performance.**

***Index Terms*—Content distribution, naming system, content routing, distributed system.**

## I. Introduction

The Internet connects billions of users across the planet. However, the changes of customer requirements cripple many of the fundamental designs of the Internet. Most importantly, the Internet has evolved into a global content sharing platform, which poses many requirements that are not addressed by the original objectives of the Internet. Another important trend is that people are increasingly using mobile devices to access content on Internet. The prediction is that mobile content consumption will become a significant part of the Internet content sharing. In this paper, we use the term *dynamic networks* to refer to such networks that involve mobile devices accessing content through the Internet. Host and content mobility is a norm in dynamic networks, but they are difficult to support in the existing Internet architecture.

In this paper, we present the design and evaluation of a scalable naming system for dynamic networks. The basis of the system is a *attribute-value*-based content and host naming scheme, which is highly flexible, expressive, and extensible. In order to support host and content mobility, our system employs dynamic binding between names and addresses. We design an open architecture for distributing name resolution requests among a large number of *name servers* (NS). Specifically, the system organizes NSes into a P2P overlay network called the *name resolution network*, which stores the names and locations of the content. Every end host connects with a *name resolver* (NR) and delegates all name resolution requests to it. NRs are connected with name resolution networks. We build the name resolution networks based on Chord-based DHT networks [1]. *Semantic-aware naming* (SAN) is employed to provide the upper-layer applications an easy-to-use programming interface. Actual content delivery, after content names are resolved, is done through Internet transport layer services.

We conduct simulations using realistic network settings extracted from real network topology data [2]. Simulation results prove the applicability of the system and demonstrate its performance. Our primary contributions in this paper are as follows:

- A novel design of a naming system for content distribution in Internet-scale dynamic networks.
- A suite of protocols and algorithms that achieve fast and scalable name resolution and content distribution.
- A feasibility analysis of the proposed system and extensive simulation studies.

The rest of this paper is organized as follows: Section II presents our naming scheme. Section III discusses our name resolution algorithms. Section IV elaborates on the transport protocol for content delivery. Section V analyzes the feasibility of the implementation of our proposed system. Section VI evaluates our design using simulations. Section VII presents relevant previous works. Section VIII concludes this paper.

## II. Content and host naming

### A. Content naming

A content name in our system contains three parts: *name modifiers* (NMs), *performance cues* (PCs), and a *tag*. We elaborate on each of them in the following sections.

*1) Name modifier:* A name modifier is a *semantic aware name* (SAN) of the content object. An NM is a string enclosed by a pair of square brackets. The string is a *Boolean expression* (BE). A BE consists of three parts: an attribute name, an attribute value (or two values), and an operator. For example, "[height, 180, $\leq$]" describes that the "height" attribute has values less than or equal to "180." Using BEs facilitates efficient content aggregation. Based on the operators used in NMs, they are classified into *equality NMs* (a "=" operator) and *aggregate NMs* ($<$, $>$, $\leq$, $\geq$, IN). Here, IN means inside a range.

*2) Performance cue:* Inspired by the work in [3], we introduce *performance cues* (PCs) in content name. A PC has the same format as a normal NM. It is indicated by a "." symbol at the beginning of the NM string. A PC is a semantic interface exposed to the upper-layer application by the naming system, which grants the applications the capability to specify certain performance requirements. For example, "[., delay, 100, <]" indicates that the delay should be less than 100. PCs are not mandatory but directive, since our system is built on top of IP networks, which only provide "best-effort" service. The underlying protocols, name resolvers, and content routers are free to choose appropriate actions based their local policies. Details on PC processing in name resolution algorithm are presented in Section III.

*3) Tag:* A tag is a semantic-free bit signature that uniquely identify the content object itself. As discussed in previous works [4], cryptographic hash function is used to obtain a 128-bit value for a content object. This tag is a unique identity of the data itself. It can be used in various scenarios to provide additional performance gains. For example, a cache management service can use tag to facilitate fast lookup; and the integrity of contents is provable by verifying its tag. Tag is not used in name resolution. The reasons to prefer NMs, which are semantic-aware, to Tags are as follows:

- SFN/SFR causes semantic mismatch between the content name and the underlying storage details. The same object may be stored in different formats, which results into distinct tags. They will be treated as multiple objects in SFN/SFRs. However, such differences are really not crucial for users to access the content. For example, the same movie may be encoded using different video codec, whereas the viewers are only concerned the visual presentation.
- SFN/SFR are difficult to understand and interact with for users. A opaque binary string does not help users decide what to do with the content it refers to.
- SANs are more flexible than SFN/SFRs. SANs can be converted to SFN/SFRs, for example, through a hash function. But it is impossible to infer SANs from SFN/SFRs.

Nonetheless, SFN/SFRs are considered more efficient for machines to process. This is why they are included in the content naming. But our system works with SANs most time. The use of SFN/SFRs is left for potential future work. The number of TAGs is equal to that of the content objects in the whole network. For the reason mentioned above, the number of names is much less than that of TAGs. This fact reduces the hardware requirements to implement the system. A realistic analysis on the feasibility of the system is in Section V.

### B. Host naming

We borrow the design of DNS in our host naming scheme. The details can be found in relevant standards [5], [6] and are omitted here. In a fully-qualified content name, the host name is treated as a literal string name modifier, although its processing method is unique. For the fast processing of names, the host name should be the first NM. In order to eliminate the ambiguity with a normal literal string NM, we use double brackets to enclose a host name: "[[host name]][$NM_1$]...". Note that host name can also be an IP address. In case the user prefers to obtain content from a provider, he/she can put the name of that provider in the content name he/she is interested in. For example, Alice wants to find news about Libyan's revolution from "cnn.com", she will coin a name "[[cnn.com]][Libyan revolution][type,=,text]".

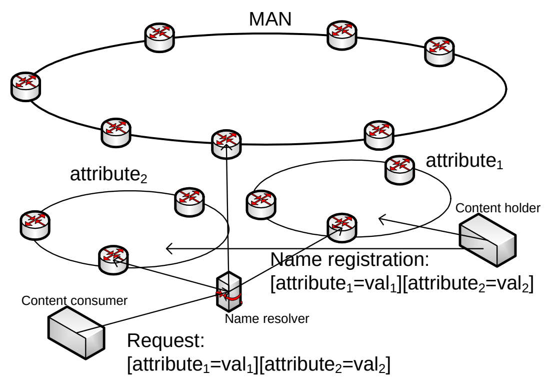

Fig. 1. The overview of our name resolution system. There is a two-level hierarchical indexing system organized around attributes and their metadata. Content holder registers the names of their contents in to the corresponding ARNETs. User requests a content object by its name. The name resolution network returns the hostname and the corresponding IP address to the requester.

## III. Name resolution

### A. Attribute-wise resolution network

Fig. 1 depicts the overall architecture of our naming system. The whole system has two levels: an *attribute meta network* (AMNET) at the higher level; and multiple *attribute-wise resolution networks* (ARNETs) at the lower layer, which store the *name records*. Both AMNET and ARNET require a key-value storage service. The following sections present the details of these two networks. An ARNET for attribute "X" indexes name records according to their values of attribute X. Thus, if a name record does not define attribute X, it will not exist in this ARNET. There are the same number of ARNETs as the number of attributes defined in the system. They are not required to be physically separated, i.e. a name server may join multiple resolving networks. As a result, every ARNET contains only a fraction of the name records in the system, which balances the workload. The union of the name records on all ARNETs consummates the name records in the entire name space. This scheme requires that every name record must be replicated $|NM|$ times over $|NM|$ name resolving networks for $|NM|$ number of name modifiers of the name record. This will potentially require a huge amount of storage space. The impact of this design on the overall practicality and performance of the system is presented in Section V.

When a name resolution request is issued from an end-host, it is first sent to its connected *name resolver* (NR). The end-host delegates its name resolution to the NR and waits for the

**Algorithm 1** Name resolution algorithm

```
1: INPUT: $name$ a name needs to be resolved;
2: OUTPUT: A host name and an IP address;
3: $ATTR_{name} \leftarrow$ the attributes of $name$;
4: $ATTR_{ARNET} \leftarrow$ the attributes of connected ARNETs;
5: $ATTR_{match} \leftarrow ATTR_{name} \cap ATTR_{ARNET}$;
6: if $ATTR_{match} = \phi$ then
7:     $ATTR \leftarrow$ a random attribute from $ATTR_{name}$;
8:     Query AMNET to get available name servers for $ATTR$;
9:     Delegate the query of $name$ to the name server with the smallest RTT to the name resolver;
10: else
11:     for Each attribute $attr$ of $ATTR_{name}$ do
12:         Obtain the hash value of $attr$;
13:     end for
14:     Find the name server $s$ that has the minimum distance to the location pointed by the hash value of $attr$;
15:     Forward the request to $s$;
16: end if
17: if Query failed; then
18:     Try the next available name server with the minimum distance to the hashed location;
19: end if
20: Return the response obtained from $s$;
```

response. A NR accesses multiple ARNETs by connecting to the name servers of the ARNETs. As shown in Fig. 1, a host named "no1.ccm" connects with a name resolver, which are connected with two name servers of two distinctive ARNETs. An ARNET contains *name servers* (NSes) that store name records. The NR forwards the request to the ARNET that has the minimum latency by sending the resolution request to the corresponding name server. As discussed above, name records are indexed according to their NMs. NMs can be literal strings or BEs. We use ChordDHT [1] to organize ARNETs. There are different types of NMs that require different treatment when storing them into the ChordDHT.

- String-valued equality NMs. A string hash function provided by the DHT of its associated ARNET is applied.
- Numerical-valued equality NMs. The entire value range is distributed on the identifier space of the DHT. The mapping is provided as default in ChordDHT.
- Numerical-valued range NMs. These refer to the NMs with operators $<$, $>$, $\leq$, $\geq$, IN. Follows the same value-space-to-identifier-space mapping presented above, such NMs are replicated on all name servers of which associated value range overlapped with the range specified by the NMs.

## B. Meta network

An NR forward a name resolution request to the ARNET(s) that it is aware of. Such information is configured by a human network administrator. If the NR has no connected ARNET for any of the attributes defined by the resolution request, the portal will consult another network to access the appropriate ARNETs. This network is called *attribute meta network* or AMNET. An AMNET is also a distributed database. However, it stores the IP addresses of the name servers of every ARNET.

| Name | Meaning |
|---|---|
| $num_{name}$ | the total number of names present in the system |
| $num_{attr}$ | the total number of attributes present in the system |
| $avgnum_{attr}$ | the average number of attributes per name in the system |
| $k$ | the maximum number of ARNETs a name can replicate |
| $num_{ns}$ | the total number of name servers in the system |
| $num_{ns}^{attr_i}$ | the total number of name servers for attribute $attr_i$ |
| $num_{ns}^{attr}$ | the average number of name servers per attribute |
| $avgnum_{arnet}$ | the number of ARNETs connect to each name resolver |

TABLE I
SYMBOLS USED IN THE PERFORMANCE ANALYSIS OF ALGORITHM 1

The stored data looks like "[attribute, IP addresses]". The key is the attribute name, the value is the IP addresses of all name servers on the ARNET of that attribute. There is only one AMNET on the entire Internet. It handles request like: "what machines stores name record information for attribute X", and returns a set of IP addresses of the machines of the ARNET for attribute X. In our design, the AMNET should be managed by a independent global organization like ICANN; and various ARNETs are managed by individual participating organizations and ISPs.

## C. The name resolution algorithm and its performance analysis

We list the pseudo code for the name resolution algorithm in Algorithm 1. Its complexity is analyzed in this section. The metric we consider is the hop count traversed on ARNETs for a end-user to resolve a content name. First, we define the notations used in the analysis in Table I. Suppose every name resolver connects to the most popular ARNETs for the end-users connecting to it. Suppose that the probability of every attribute appears in every name follows a uniform distribution, then for each attribute, the number of names containing that attribute is $num_{name} \times \frac{avgnum_{attr}}{num_{attr}}$. The total number of names in the system is $avgnum_{attr} \times num_{name}$ because names are replicated on ARNETs associated with its attributes. So each ARNET should allocate a proportional number of name servers, which is given by the following equation:

$$
\begin{aligned}
num_{ns}^{attr_i} &= \frac{num_{name} \times \frac{avgnum_{attr}}{num_{attr}}}{avgnum_{attr} \times num_{name}} \times num_{ns} \\
&= \frac{num_{ns}}{num_{attr}} \qquad (1)
\end{aligned}
$$

That is, all name servers are equally allocated for all ARNETs. So the average query hop-count for each ARNET is $log_2(\frac{num_{ns}}{num_{attr}}) = log_2 num_{ns} - log_2 num_{attr}$. The above analysis is valid only for the case where $k \geq avgnum_{attr}$. If $k < avgnum_{attr}$, Eq. 1 becomes:

$$
num_{ns}^{attr_i} = \frac{k}{avgnum_{attr}} \times \frac{num_{ns}}{num_{attr}} \qquad (2)
$$

## IV. CONTENT DELIVERY

## A. Basic content transport scheme

The control logics of the content delivery is separated from transport layer details. An end-host sends the name resolution

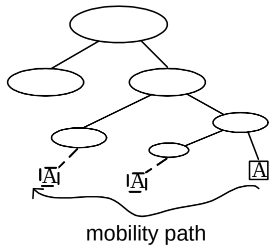

Fig. 2. Selecting a gate keeper for mobile host "A" in a tree topology. It is straightforward to find a content router that preserve the shortest path on a tree topology, i.e. by selecting the lowest common ancestor of the possible leaves where the host may reside.

request to the system, which is processed as discussed above. After obtained the address of the content holder, the end-host send requests in a separate channel to the content holder. Host mobility is handled by a novel idea called *mobility realm* (MR). A mobile host *belongs to* a MR, if the host can be addressed at any time by a static entity called *gate keeper*. Every mobile host must register its address to the gate keeper of the MR it resides. The intention is to guarantee the routing performance: for any MR, the shortest path connecting any host outside of the MR and any mobile host within the MR contains a gate keeper. For example, as depicted in Fig. 2, select the lowest common ancestor of all the addresses that the mobile host is possible to reside in the hierarchical address tree. Gate keeper is the key to efficiently handling content mobility and host mobility. A mobile content holder reports its host name and the IP address of the gate keeper (remember that it is statically addressed). The information is then sent with the request to the gate keeper. The gate keeper then obtains the current IP address of the content holder by looking up its local address table, and forward the request to the host. Similarly, a mobile host needs to report its name and the IP address of its gate keeper, so the content holder can dynamically forward the content back to a mobile host.

### B. Load balance

Our load balance scheme is based on dynamic name resolution. In the first place, our naming system has the ability to return appropriate results when multiple hosts providing the same contents, which provide a effective way for content holders to deploy distributed content store system that spread the load. Our scheme, on the contrary, is centered on the dynamic name resolution. There are an overlay of *content routers* (CRs). They are situated over the underlying IP networks. CRs do not directly possess data but only forward and cache content sent from end-hosts. When an end-host is overloaded, it informs a down-stream CR and then requests the CR to help itself forward part of its content. The CR achieves this by registering itself as the content holder of the data the end-host wants to migrate to the ARNET. This registration has a TTL. The logic is that the CR is for transient load migration. A TTL ensures that the registered contents get purged from ARNETs to avoid a persistent load on CRs. If a end-host is constantly overloaded, the content provider should deploy more machines instead of relying on the load balancing feature.

## V. Feasibility

### A. Name server

We use the number of web pages to estimate the number of name records. The estimated count of indexable web page is $11.5 \times 10^9$ as of 2005. Since these objects can be aggregated, the actual name counts can be noticeably less. We assume that the number of name records is $10^{10}$. A name record contains a name, a host name, and an IP address. According to the measurement study in [7], the average URL length ranges from 62 to 81 for 8 countries. We use 81 bytes as the average URL length, and a 128-bit tag is included in every content name. So the average content name size is 97 bytes. We obtained the average length of the domain names of the top 1 million sites listed on Alexa [8] to 14.8 bytes. The size of an IPv4 address is 4 bytes. So the average size of a name record is the sum of these three parts, which is 116 bytes. The size of all name records on Internet is $10^{12}$ bytes. Suppose each name record is replicate for 5 times. If we assume 10% of them are frequently requested and should be put into RAM, then the required RAM size is $5 \times 10^{11}$ bytes. Assuming that a machine has 16GB or $16 \times 10^9$ bytes, less than 50 machines is enough to store all name record in RAM. Therefore, memory is not a limiting factor for performance or cost. We assume that the rate of the name resolution request will be in the same order as the HTTP request. We use the analysis results given in [9]. Each peripheral name server handles 20,000 requests per second. If 8 core machines are used, each core needs to process 2,500 requests per second. Each request consumes 400 microseconds. This is considered an achievable requirement according to the analysis in [9]. The required bandwidth is 16Mbps given an average name record size of 116 bytes, which is a modest requirement. Suppose each named content has a valid lifetime of 1 week. In 1 week, all name records are refreshed and need to register again. The equivalent name registration rate is 16,535 names per second. So the estimated registration bandwidth cost is 15.3Mbps.

### B. Content router

Content routers' use of hardware and network resources is hard to predict since it depends on the sizes of content objects and underlying transport protocols. According to the statistics of Google [10], the average size of web pages is 320KB. This is about 3,200× of the size of the name records on Internet. Since content delivery requires lower response rate than the name resolution, disks can be used. The disk volume of the existing commercial PCs can have a 1,000× space of the RAM. Suppose each content object needs to be replicated for 10 times, the required machines will be 32 times of the name servers. It is even more difficult to predict the bandwidth use of content routers. The only plausible argument is that most CDN service providers can make profits from their business, therefore, it is reasonable to predict that by deploying such infrastructure of content routers, companies can make

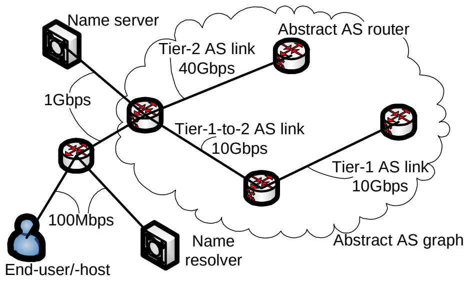

Fig. 3. The illustration of the simulation settings. An abstract AS graph is extracted from a detailed router-level graph. End hosts are wired with routers of every AS. A name server and a name resolver is allocated for each AS.

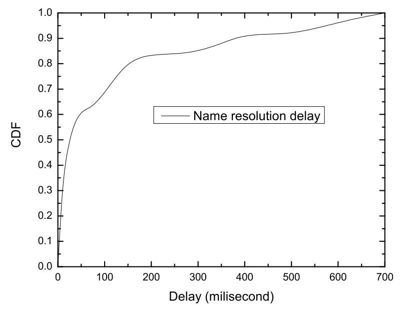

Fig. 4. The CDF of the name resolution delay.

profit from content holders who are willing to provide better performance to their users.

### C. AMNET

Recall that AMNET is a globally-visible distributed database of the servers location information of the ARNETs. The required storage space will be much smaller than the ARNETs. Since name servers are contributed by individual organizations, their IP addresses are generally stable, so the registration load on AMNET should be easy to handle. The query load, however, can be quite overwhelming, since any name resolver can initiate queries if it has not sufficient knowledge of the ARNETs. A solution would be letting all name resolvers maintain connections to at least one name servers of all ARNETs. If the total number of attributes present in the system is small, this can be a viable solution. But it is generally believe that the interests of Internet users follow a long-tail distribution, which means a large number of attributes are necessary to accurately describe the majority of user demands. Another possible, solution will be using caching on name resolvers to reduce unnecessary queries to AMNET. Similar techniques are used in CoDNS [11]. We deem AMNET is best deployed and maintained by a global third-party organization, like ICANN. Considering the relatively low hardware requirements, this makes most sense from a economic and political point of view.

## VI. Performance evaluation

### A. Simulation settings

The Internet topology data from CAIDA data trace is used to extract a realistic network setting. The data set contains a router-level topology of over 3 million routers are collected. More details of the dataset are available at CAIDA's website [2]. We extract the AS-level topology from this topology, with the help of a router-to-AS mapping data. We model the link delay between ASes in three parts: queuing delay, transportation delay, and propagation delay, which are determined by the routers' processing power, link bandwidth, and geographic distance between end-routers, respectively. The link delay is shown in Fig. 3 and the propagation delay is derived from the geographical distance between ASes. A machine of 16GB memory and 8-3GHz core CPU is used as name server and content router. We perform a measurement of the processing speed on the machine. The result shows that the average processing time is about 100ms.

Each end-host populates 10 name records. The total number of end-hosts is 1,703,858, and the totally number of name records in the system is 17,038,580. Each name record has 5 attributes, of which 2 are string valued and 3 are numerical valued. A end-host is chosen uniformly to an existing name record in the system. We perform 10,000 such requests and record their delays, and plot a CDF figure of the obtained data. To measure the content delivery throughput, we randomly chose 100 end-hosts to request the same content.

### B. Name resolution delay

Besides the delay incurred by network transport, another important source is the internal processing speed on namer servers. This metric is closely related to the size of the stored elements. We employ a hash-table. Based on the analysis in Section V, the total number of name records is $10^{10}$, and the total number of ASes in the system is 18,810, the average number of name records per name server is 531,632. The results of simulation has small deviations due to imbalanced loads. The delay of name servers looking up the name record is measured by the actual running time of the corresponding code, on a 3GHz Intel CPU. The CDF of the delay of name resolution is given in Fig. 4. The delay is measured as the time duration from the request is issued to the results are received by the requester. The measured largest delay is around 700ms, which is resultant from the inter-continental paths. We found that the processing time on each name servers is a major source of the end-to-end delay. The lookup in DHT networks is very sensitive to the placement of data. This set of simulation is insufficient to evaluate its impacts since the settings are much simplified. This will be addressed in our future work. The average value is 275ms. Overall speaking, the performance is quite appealing. Note that we have not applied any optimization techniques here, like exploiting query locality or caching.

### C. Content delivery throughput

TCP is used for the transport between end-users and content-holders, and content routers. Congestion control and reliability follows the basic TCP designs, certainly it cannot

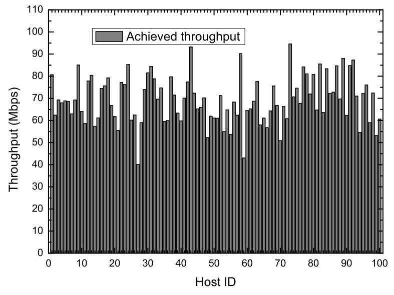

Fig. 5. The achieved throughput of all hosts.

compete with the optimized implementation in modern operating systems. A 100MB file is sent to all requester, we plot the achieved bandwidth for all the 100 requesters in Fig. 5. We conclude that the content delivery scheme achieves sufficiently good results. The access link bandwidth is 100Mbps. However, we would like to point out that the simulation settings for throughput measurement is extremely simplified from the real Internet environment. In the future work, we will evaluate the scheme in a testbed environment.

## VII. Related work

In [12], the authors study the impact of DNS lookup on the performance of web content retrieval. Their findings show that DNS lookup delay contribute a significant part of the content delivery delay, which proves the importance of fast name resolution systems. Semantic free reference (SFR) [4] is a proposal to use fixed-length bit-strings to index all web objects. A global DHT-based P2P network is used to store and indexing all SFRs. Our system differs from SFR in two important ways: our use of semantic-aware names, and hierarchical indexing networks, which results into a more commercially applicable system. In [13], the authors advocate a layered naming architecture for the future Internet. Intentional Naming System (INS) [14] is another hierarchical naming protocol. Our naming scheme uses independent name modifiers, which does not enforce fixed hierarchy. Our design provides more flexibility in implementation at the cost of name record replications. INS' hierarchical routing structure is based on names, which limits the flexibility of participating organizations. CDNs utilize various existing infrastructure to facilitate efficient content delivery. Among many CDN designs, CoralCDN [15], [16] has a hierarchical P2P network to support fast name resolution. CoralCDN's P2P network is based on physical network connection speed, which opportunistically finds the closest content copy to the requester. Our design only resolves names to content holders, which dose not itself handles content copying. CoralCDN has a data transfer protocol [17] provide anycast service for content delivery from multiple content-holders. Our system is more general than CDN in the sense that it is implemented as a global infrastructure and grant individual organizations the ability to manage their own content delivery policy.

## VIII. Conclusion

We present the design and evaluation of a naming system for scalable content distribution for dynamic networks. The system aims to be a general content distribution platform for the future Internet. We designed a flexible and extensible naming scheme based on independent name modifiers. To achieve fast name resolution, we employ a novel hierarchical name resolution network, which achieves fast name resolution through name record replication and provides a simple and flexible interface to application developers. Using this flexible naming system, we described how to balance system load.

## Acknowledgement

This work was partially supported by the National Natural Science Foundation of China under Grant No.60903222.

## References

[1] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, pp. 17–32, February 2003.

[2] CAIDA, "Topology research," http://www.caida.org/research/topology/.

[3] J. Stribling, Y. Sovran, I. Zhang, X. Pretzer, J. Li, M. F. Kaashoek, and R. Morris, "Flexible, wide-area storage for distributed systems with wheelfs," in *Proc. of NSDI'09*. Berkeley, CA, USA: USENIX Association, 2009, pp. 43–58.

[4] M. Walfish, H. Balakrishnan, and S. Shenker, "Untangling the web from DNS," in *Proc. of NSDI'04*. Berkeley, CA, USA: USENIX Association, 2004, pp. 17–17.

[5] P. Mockapetris, "Domain names - concepts and facilities," RFC 1034 (Standard), Internet Engineering Task Force, Nov. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1034.txt

[6] ——, "Domain names - implementation and specification," RFC 1035 (Standard), Internet Engineering Task Force, Nov. 1987. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[7] R. Baeza-Yates, C. Castillo, and E. N. Efthimiadis, "Characterization of national web domains," *ACM Trans. Internet Technol.*, vol. 7, May 2007.

[8] Alexa, "Top 1 million site list," http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[9] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192, August 2007.

[10] Google, "Web metrics: Size and number of resources," http://code.google.com/speed/articles/web-metrics.html.

[11] K. Park, V. S. Pai, L. Peterson, and Z. Wang, "CoDNS: improving dns performance and reliability via cooperative lookups," in *Proc. of SOSP'04*. Berkeley, CA, USA: USENIX Association, 2004, pp. 14–14.

[12] C. E. Wills and H. Shang, "The contribution of dns lookup costs to web object retrieval," WPI technical report, 2000.

[13] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, "A layered naming architecture for the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 343–352, August 2004.

[14] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," *SIGOPS Oper. Syst. Rev.*, vol. 33, pp. 186–201, December 1999.

[15] M. J. Freedman, "Experiences with coralcdn: a five-year operational view," in *Proc. of NSDI'10*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 7–7.

[16] M. J. Freedman, E. Freudenthal, and D. Mazières, "Democratizing content publication with coral," in *Proc. of NSDI'04*. Berkeley, CA, USA: USENIX Association, 2004, pp. 18–18.

[17] M. J. Freedman, K. Lakshminarayanan, and D. Mazières, "Oasis: anycast for any service," in *Proc. of NSDI'06*, ser. NSDI'06. Berkeley, CA, USA: USENIX Association, 2006, pp. 10–10.