HAL
archives-ouvertes.fr

A Network Aware Resource Discovery Service

Luigi Liquori, Rossano Gaeta, Matteo Sereno

▶ **To cite this version:**

Luigi Liquori, Rossano Gaeta, Matteo Sereno. A Network Aware Resource Discovery Service. EPEW 2019 - 16th European Performance Engineering Workshop, Nov 2019, Milano, Italy. hal-01895452v3

HAL Id: hal-01895452
https://hal.inria.fr/hal-01895452v3

Submitted on 12 Mar 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Network Aware Resource Discovery Service

Luigi Liquori[1][0000−0003−3961−4205], Rossano Gaeta[2][0000−0002−6521−403X], and Matteo Sereno[2][0000−0002−5339−3456]

[1] Université Côte d'Azur, INRIA Sophia Antipolis - Méditerranée, France
[2] Università di Torino, Dipartimento di Informatica, Torino, Italia

Abstract. Internet in recent years has become a huge set of channels for content distribution highlighting limits and inefficiencies of the current protocol suite originally designed for host-to-host communication. In this paper we exploit recent advances in Information Centric Networks in the attempt to reshape the actual Internet infrastructure from a host-centric to a name-centric paradigm where the focus is on named data instead of machine name hosting those data. In particular, we propose a Content Name System Service that provides a new network aware Content Discovery Service. The CNS behavior and architecture uses the BGP inter-domain routing information. In particular, the service registers and discovers resource names in each Autonomous System: contents are discovered by searching through the augmented AS graph representation classifying ASes into customer, provider, and peering, as the BGP protocol does.
Performance of CNS can be characterized by the fraction of Autonomous Systems that successfully locate a requested content and by the average number of CNS Servers explored during the search phase. A C-based simulator of CNS is developed and is run over real ASes topologies provided by the Center for Applied Internet Data Analysis to provide estimates of both performance indexes. Preliminary performance and sensitivity results show the CNS approach is promising and can be efficiently implemented by incrementally deploying CNS Servers.

Keywords: Discovery Service, Naming, Performance evaluation, Network and economical awareness.

1 Introduction

Information Centric Networks (ICN) is a clean-state approach to redesign the actual Internet infrastructure from a host-centric, fully connected, paradigm to a name-centric, loosely connected, paradigm where the focus is on named data instead of machine name hosting those data. In the last decade many proposals raised from research to capture this new paradigm: they mainly can be grouped into two schools of thought: *Content Centric Networks* referring to the Jacobson-based vision [6,11,13], where routing is driven by fully qualified - human readable - hierarchical names and *Data Oriented Network Architecture* referring to a flat, unreadable but unique name-space [7] (see also [1, 3, 12]).

While it is always exciting to conceive a new network starting from new concepts and from a clean-state design, network's history teaches us that the Internet infrastructure and its protocol suite have little changed; this is, quite obviously, because of strong backward-compatibility needs, and because of the tremendous expansion of the Internet phenomenon.

This paper supports the evolutive research line and presents a lightweight network aware Internet Service to be implemented between the Transport and the Session layers (referring to the ISO-OSI protocol layering). We call this new service *Content Name System* (CNS) organized throughout a set of communicating CNS Servers and we design a protocol, called `link`, implementing the Service Discovery.

The purpose of this Discovery Service is to publish machine-IP-addresses being the owners (or the purveyors) of some named-contents and retrieve that machine-IP-addresses performing a distributed search using the named-content as the database-key. The service binds a set of IP-addresses to content-names, the latter referred by *hypernames*. The CNS Service stops when some or no IP-addresses are returned or when no other CNS Server can be delegated in the iterative call implementing the distributed data-base query. Each CNS Server is equipped with a database containing for each queried content-name, the set of corresponding IPs ordered by local awareness.

In our proposal, the CNS Servers are distributed over the Internet *Autonomous Systems* (AS): there is one CNS Server per AS (or for load balancing purposes there can be multiple CNS Servers) taking care of resources registered inside the AS itself. Furthermore, the Discovery Service leverages on AS relationships by mimicking their hierarchy; in the CNS Service each AS uses the (business) relationships with the ASes in its neighborhood to also drive the content location process according to the so called "valley-free" property, i.e., that the discover process does not generate any supplementary cost for the AS involved in the discovery. Therefore, the main contributions of the paper are:

- The definition of a Resource Discovery Service (CNS) with related protocol `link` allowing to search contents names through Autonomous Systems: the service is achieved by defining *i)* a naming notation to denote contents, *ii)* a distributed database implemented by CNS Servers deployed along ASes, and *iii)* the `link` Discovery Protocol, to route queries along the ASes.
- The development of a C-based simulator of CNS and the `link` protocol to conduct a preliminary performance and sensitivity analysis of the proposed CNS. To this end, performance of the proposed CNS is represented by the *hit probability* (that is defined as the fraction of ASes that successfully locate a requested object) and the *average lookup length* representing the average number of CNS Servers explored during the search phase. Experiments are run over real ASes topologies provided by CAIDA [2] to provide reliable and meaningful estimates of both performance indexes; we also analyze how high performance can be if Tier-1 ASes are excluded from the search phase providing that Tier-2 ASes are incentivized to cooperate.

The rest of the paper is organized as follows: in Section 2 we define the proposed CNS (we define hypernames, the CNS Servers, and the `link` protocol); Section 3 presents preliminary simulation results to assess the performance of CNS Service. Finally, Section 4 summarizes the paper contributions and discusses some further developments.

2 Content Name System

2.1 Hypernames

A *hypername* (HN) is a human readable string denoting the content name, enriched with a number of optional parameters to identify its ownership, its integrity, its hosting, and its attribute-list. Hypernames are generated by the following abstract syntax:

`[fing_princ:][fing_cont:][hosts:][tags:]cont_name,` where

- `cont_name` is a (possibly human readable) string denoting a content name (e.g. "openoffice.iso", "traffic_light", "defibrillator", "plastic_bottle", "pedestrian", URI, MAC, GUID, etc.);
- `tags` is an optional (possibly human readable) list of keywords (e.g. "sell", "buy", "rent", "cars", etc) associated with a given content;
- `hosts` is an optional list of hostnames being the purveyors of the content: when a hypername contains a list of hostnames, then the content name is retrieved from one of the hostnames: the local CNS perform a DNS query, transforms one (or all) hostname(s) into IP address(es) and return that list to the sender of the discovery request;
- `fing_cont` is an optional digital signature (hash) denoting the integrity of the content to be retrieved;
- `fing_princ` is an optional digital signature denoting the public asymmetric key of the principal, i.e. the owner of the content: it allow to identify the identity of the latter as soon as we retrieve the content itself.

Therefore, a hypername is characterized by a human readable part and another part which is human unreadable.

2.2 CNS Servers

Similarly to the well-known DNS Service, we locate the CNS Server in the ISO-OSI hourglass at the application level. The goal of CNS is to translate hypernames into lists of IP addresses, that is $\mathrm{HN} \Longrightarrow \{\mathrm{IP_i}\}^{\mathrm{i}\in\mathrm{I}}$ with $\mathrm{I} = \emptyset$ in case of discovery failure.

BGP Business Relationship Among ASes. CNS servers are distributed over the ASes; more precisely, there is one CNS per AS (or for load balancing purposes there can be multiple servers) taking care of resources registered inside the AS itself. The Discovery Service leverages on AS relationships: the CNS server hierarchy mimics the AS relationship hierarchy. It is well-known that the routing between ASs (also called interdomain routing) is determined by the *Border Gateway Protocol* (BGP) [10]. The main feature of the interdomain routing is that it allows each AS to choose its own administrative policy in selecting the best route, and announcing and accepting routes.

The commercial agreements between two ASes domains can be classified into two main classes of agreements: *customer to provider*, and *peering*[3]. An AS customer pays its AS provider (or ASes in case of multiple providers) for connectivity to the rest of the Internet. A pair of ASes can set up a peering relation and in this case they agree to exchange traffic between their respective customers free of charge.

The AS-graph annotated with these two kinds of relationships is one of the most famous and studied representations of the Internet (e.g., see the measurement studies provided by CAIDA [2].) In this graph, according to their roles, we can distinguish three different kind of ASes: Tier-1, Tier-2, and Tier-3. A Tier-1 is an AS that can reach every other destination on the Internet without paying other ASes. In other words, a Tier-1 is an AS with (many) customer ASes but with no provider. For connectivity purposes, the Tier-1-s set up peering relationships among them. On the other hand, a Tier-3 is a stub AS, without any transit customers, and with some peering relationships. Tier-3 ASes generally purchase transit Internet connection from Tier-2 ASes and, in some cases, even from the Tier-1 ASes as well. Finally, a Tier-2 is an AS with customers, and some peering, but that still buys transit service from Tier-1 ASes to reach some portion of the Internet.

The relationships among the ASes play a fundamental role in shaping the AS graph structure and in defining the routing policies implemented throughout BGP. In particular, the paths between two ASes must avoid routing policies that would result in unjustified payments by some AS. Examples of such incorrect routing paths are, for instance, an AS provider that routes the traffic directed to another AS provider by forwarding it to one of its AS customers. This path is incorrect because would cause an unjustified cost in charge to the AS customer used as an intermediate. Another example of incorrect path occurs when an AS forwards its traffic by using as intermediate step one of its peering relationships. In this case the peer AS chosen as intermediate would be in charge of the transit cost for the traffic it forwards. Figure 1 (presented in [5]) shows a simple configuration of seven ASes, their relationships (i.e., provider-to-customer and peering). On this simple AS graph we report two wrong, and two correct paths.

The routing paths among pairs of ASes is obtained by the BGP protocol that uses selective exporting path rules (i.e., each AS selectively provides transit

[3] The ASes can establish also other type of relationships such as "sibling" and "backup". For the purposes of this paper we neglect them.

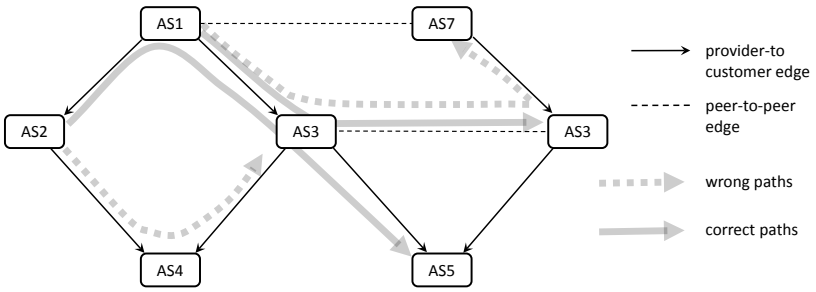

Fig. 1. A simple AS graph with two wrong and two correct routing paths.

services for its neighboring ASes). All the paths (also called routes) with property we previously discussed are called *free-valley* (or *no-valley*) paths [5].

CNS Hierarchical Topology. The CNS distributed database is organized into a hierarchy of CNS servers deployed according to the Tier-1/Tier-2/Tier-3 AS topology. Each AS must have at least one CNS, called *authoritative*, whose database will take into account the association of each hypername with a list of IPs that have registered a content named by a hypername. The authoritative CNS also knows exactly its position in the distributed database, namely *i)* the IP addresses of all customers' CNS, *ii)* the IP addresses of all providers' CNS and *iii)* the IP addresses of all peer-to-peer CNSs: this will allow to dispatch queries along the distributed database.

In order to make a content discoverable, the owner or purveyor publishes an hypername referring to a content in the local CNS. Note that the publication in a CNS associates the hypername with a principal, and that principal holds the content as an owner or a purveyor (the content being mutable or immutable). Suppose a given content be available by a host belonging to an autonomous system: the host can publish, through the CNS Service, the content in the authoritative CNS local database. To do this, at the beginning, the host creates a proper hypername that will be sent as a formal parameter to the authoritative CNS. Note that the host decides which attribute to attach to the hypername and if it should publish that content as a owner or as a purveyor. In the first case (owner) the publication is done by a simple write in the CNS' database[4]. In the second case (purveyor) the host could be asked to package a `.torrent` file and write it in the CNS' database. Following the Bittorrent jargon, the purveyor plays a role of seed and it will be asked to publish itself as a purveyor of the content every time interval: further nodes entering the swarm for the content will

[4] Depending on a local policy, the CNS could ask to republish the content every n seconds.

```
1.01 on receipt of link(HN,DOWN) from provider do  // receive a query from a ''downhill"
1.02 value = lookupdb(HN);  // search HN in the CNS' local data base
1.03 if (value ≠ 0) // some IP publishing HN are found
1.04 then {publish(HN,value) to CNS; return value to provider};  // write in the local CNS
                and return IPs ''back to the downhill''
1.05 else list = select(α,customerlist);  // select some customers CNS
1.06 forall cus ∈ list do value = value ∪ send link(HN,DOWN) to cus;  // and forward the
                query downhill through a customer
1.07 publish(HN,value) to CNS; return value to provider;  // write in the local CNS and
                return IPs ''back to the hill''
```

Fig. 2. Queries from provider with downhill direction continue on α thread downhill.

```
2.01 on receipt of link(HN,UP) from peer do  // receive a query from a peer on the ''top of the hill"
2.02 value = lookupdb(HN); // search HN in the CNS' local data base
2.03 if (value ≠ 0) // some IP publishing HN are found
2.04 then {publish(HN,value) to CNS; return value to peer}; // write in the local CNS and
                return IPs ''back to the top of the hill''
2.05 else list = select(α,customerlist); // select some customers CNS
2.06 forall cus ∈ list do value = value ∪ send link(HN,DOWN) to cus;  // and forward the
                query but downhill through a customer
2.07 publish(HN,value) to CNS; return value to peer;  // write in the local CNS and
                return IPs ''back to the top of the hill''
```

Fig. 3. Queries from peer with uphill direction will change on α thread downhill.

be asked to publish his name in the torrent; for that content, the CNS server would serve as a kind of network aware Bittorrent tracker.

2.3 The `link` Discovery Protocol

Each autonomous system holds an authoritative CNS server, that records the mappings for all the hypernames published inside it. The CNS database is organized hierarchically following CAIDA's augmented graph [2]. Let α, β and γ being AS-specific parameters; in a nutshell, the `link` protocol proceeds intuitively as follows:

1. the client first contacts its authoritative CNS and then searches the hypername in the local publications (i.e. in the current and in the peering CNS);
2. if the above fail, then the authoritative CNS forwards the query through α-CNS belonging to ASes in "downstream", i.e., with which we have signed some provider-to-customer agreement;
3. if the above fails, then the authoritative CNS forward the query through γ-CNS belonging to ASes in peer, i.e., with which we have signed some peering-to-peering agreement;
4. if all of the the above fail, then the authoritative CNS forward the query through β-CNS belonging to ASes in "upstream", i.e., with which we have signed some customer-to-provider agreement.

The `link` pseudocode is presented in Figures 2, 3, and 4. A client sends a query to the local authoritative CNS server, with argument the hypername `HN` and a direction `UP` (from customer-to-provider or from peer-to-peer) or `DOWN` (from

```
3.01 on receipt of link(HN,UP) from customer do  // receive a query from a ''uphill''
3.02 value = lookupdb(HN); // search HN in the CNS' local data base
3.03 if (value ≠ 0) // some IP publishing HN are found
3.04 then {publish(HN,value) to CNS; return value to customer}; // write in the local CNS and
              return IPs ''back to the uphill''
3.05 else list = select(α,customerlist); // select some customers CNS
3.06 forall cus ∈ list do value = value ∪ send link(HN,DOWN) to cus;  // and forward the
              query but downhill through a customer
3.07 if (value ≠ 0) // some CNS are suggested
3.08 then {publish(HN,value) to CNS; return value to customer}; // write in the local CNS and
              return IPs ''back to the uphill''
3.09 else list = select(γ,peerlist); // select some peers CNS
3.10 forall per ∈ list do value = value ∪ send link(HN,UP) to per; // and forward the query
              uphill through a top of the hill peer
3.11 if (value ≠ 0) // some CNS are suggested
3.12 then {publish(HN,value) to CNS; return value to customer}; // write in the local CNS
              and return IPs ''back to the uphill''
3.13 else list = select(β,providerlist); // select some provider CNS
3.14 forall pro ∈ list do value = value ∪ send link(HN,UP) to pro; // and forward the query
              uphill through a provider
3.15 publish(HN,value) to CNS; return value to customer // write in the local CNS and return
              IPs ''back to the uphill''
```

Fig. 4. A query from customer with uphill direction will continue on three directions: first α-downhill, then γ-downhill, and finally β-uphill.

provider-to-customer). This query is recursive and the client will be blocked until the CNS will answer positively with a result containing a set of addresses $\{\mathrm{IP_i}\}^{\mathrm{i}\in\mathrm{I}}$ associated with `HN`, or with a search failure.

Location process start. A client sends a query to the authoritative CNS server where the client belongs to, with argument the hypername `HN`. In DNS jargon, this query is recursive i.e., the client will be blocked until the CNS will answer positively with a result containing a set of addresses $\{\mathrm{IP_i}\}^{\mathrm{i}\in\mathrm{I}}$ associated with `HN`, or with a search failure.

Figure 2: from provider with downhill direction. This code refers to the general case when the current CNS receives a `link` message with a `HN` and a downhill direction from a provider-CNS (line 1.01). First of all, a local lookup is performed (1.02); in case of success, the result value is returned to the sender[5] (1.04); else selects α-customer-CNS (1.05) and sends α-iterative `link` queries with the same `HN` and the same downhill direction (1.06); then collects the result value and send it back to the sender of the first `link` message (1.07). Before return, all the results will be written in the local CNS in order to give a direct answer in successive queries.

Figure 3: from peer with uphill direction. Following the BGP jargon, this code refers to the case of being "on the top of the hill", i.e., receiving a message from uphill and from a peer-to-peer-CNS. Execute the same code as the one of Figure 2, with the following exception: invert the direction from uphill to downhill

[5] At the beginning of the search, the sender is just the authoritative-CNS itself, while in the middle of the location process, the sender is a provider-CNS.

when sending α-*iterative* `link` queries (2.06). Before return, all the results will be written in the local CNS in order to give a direct answer in successive queries.

Figure 4: from customer with uphill direction. This code refers to the case where a CNS receives a `link` message from uphill from a customer-CNS. Following the BGP jargon, when we receive a query from a customer and with an uphill direction the following steps are executed. First of all, a local lookup is performed (line 3.02): in case of success, the result value is returned to the sender (3.04); else select α-customer-CNS[6] (3.05) and send α-iterative `link` queries with the same `HN` but inverting the direction from uphill to downhill (push downhill the query) (3.06); in case of success, the result value is returned to the sender (3.08); else select γ-peer-CNS (3.09) and send γ-iterative `link` queries with the same `HN` and the same direction[7] (3.10); in case of success, the result value is returned to the sender (3.12); else select β-provider-CNS (3.13) and send β-iterative `link` queries with the same hypername and the same uphill direction (in other words: go uphill only after tried to invert the search downhill but all the queries failed) (3.14); as the last resort of the query, return a success or failure value to the sender. As in [6], before return all the results will be written in the local CNS in order to give a direct answer in successive queries.

Note. All α, β, and γ are dependent on the local CNS; all messages not matching with the above pseudocode are flushed by the receiving CNS server.

2.4 CNS vs DNS

Because of its resemblance with the DNS Service, we highlight differences and similarities in the following:

- DNS [9] is a fundamental phone book directory for the Internet. It mainly uses the UDP transport to query other distributed DNS servers to answer client questions like "which IP addresses are associated with the name `www.google.com`?" The DNS Service provides information about hosts querying the DNS hierarchy: this hierarchy can go through ASes and does not follow the AS cash flow route: the small amount of packets involved in DNS resolution makes DNS economically scalable. On the contrary, and this has been made explicit in the `link` pseudocode, packets will be routed following the economic interest of the AS that generates the query: this point is crucial for ensuring the Discovery Service to be economically scalable.
- DNS delegates name resolution into domain zones from the smallest to the biggest zone. With the same idea, the CNS delegates content discovery (content name resolution) through ASes always trying to follow, when possible, a reverse cash flow route in order to suggest to the further content delivery an ordinary cash flow route;

[6] Do not choose the customer-CNS that have sent the query.

[7] Successive execution of code in Figure 3 will later invert the direction from uphill to downhill, i.e. we push downhill the query.

– DNS distributed database is indexed via domain names. On the other hand, the relations among CNS servers are derived by the relations among ASes (customer-to-provider, provider-to-customer and peering relations). These relations can be derived by using CAIDA's AS relationships dataset maps (see [2] and Gao's pioneering work [5] on valley-free routing);
– DNS queries can be iterative or recursive: the same holds for CNS: nevertheless, an efficient implementation of the CNS Service prefers iterative queries.

3 Performance Results

In this section we present results that characterise the performance of the CNS Service and of the `link` protocol. Simulations show that link is able to successfully locate objects with high probability at low cost; they also show that good performance can be obtained by excluding Tier-1 ASes from the search phase providing that Tier-2 ASes are incentivized to cooperate.

Performance is represented by two indexes: the *hit probability* (denoted as p_{hit}) that is defined as the fraction of ASes that successfully locate a requested object, and the *average lookup length* (denoted as avg_{ll}) representing the average number of CNS servers explored during the search phase. To this end, we developed a C-based simulator of the proposed object Discovery Service. The simulator runs by using real ASes topologies provided by CAIDA [2] and is able to reproduce the dynamic behavior of location requests. We provide a sensitivity analysis of the lookup algorithm `link` with respect to parameters α, β, and γ. We also discuss the performance of `link` as a function of the fraction of ASes that actually deploy a CNS server to support the location service.

3.1 Scenario

In our experiments we selected an ASes topology provided by CAIDA containing all the ASes and their type of relationships. In these snapshots edges between two nodes either represent peer relationships between ASes (undirected edges) or provider-to-consumer roles (directed edges). We classify ASes in Tier-1/Tier-2/Tier-3 [2] subsets based on the topological characteristics of nodes. In particular, we define as: *i)* Tier-1 those snapshot nodes with no incoming edges, i.e., ASes that have no providers; *ii)* Tier-3 those snapshot nodes with no outgoing edges, i.e., ASes that have no customers; *iii)* Tier-2 all other snapshot nodes. We consider a resource whose popularity is equal to 0.1 among Tier-2 and Tier-3 ASes; we assume Tier-1 ASes do not hold a copy of the resource. Furthermore, we run the simulator by restricting location requests to only Tier-2 and Tier-3 ASes. To summarize, Tier-1 ASes participate in the search process but do not contribute any further.

3.2 Sensitivity to Lookup Parameters

We characterize the performance of `link` by relying only on customers, i.e., parameters β and γ are both equal to 0. In this case, Tier-3 ASes can successfully

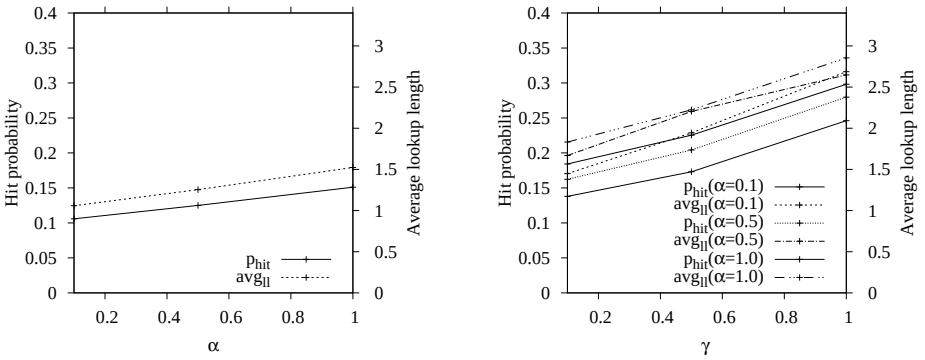

Fig. 5. Values of p_{hit} and avg_{ll} during the lookup (left plot as function of α with $\beta = \gamma = 0$, right plot as function of γ with $\beta = 0$).

resolve the location request only if they hold a copy of the resource. Tier-2 ASes can exploit more search path, instead. Indeed, their p_{hit} is higher than the resource popularity and the overall results are represented in Figure 5 (left plot). It can be noted that increasing α raises p_{hit} from 0.105902 to 0.151068: the performance for $\alpha = 1$ represents an upper bound on the achievable performance. Furthermore, the rather small values of avg_{ll} for all considered values of α show that a little number of search requests contacts more than one CNS.

To evaluate the impact peers in the AS snapshot we consider all combinations of parameters α and γ where $\beta = 0$ in results we present in Figure 5 (right plot). It can be noted that parameter γ has moderate impact since Tier-3 ASes have very limited peer AS relationships. On the contrary, Tier-2 ASes can exploit their peering relations to increase their p_{hit} although the maximum achievable performance is just 0.298151.

The impact of parameter β on the performance of `link` is remarkable, instead. It can be noted from results in Figure 6 that by increasing β from 0.1 to 0.5 for a very low value of α (0.1) we obtain $p_{hit} = 0.562925$ (from the value 0.107914). By further increasing it to 1 we obtain that location requests are successfully served almost surely for any value of α. Of course, this improvement is paid by the increased cost of the service in terms of the avg_{ll} values.

The last set of results we present is to analyze how the resource popularity impacts on the cost of lookups and how effectively `link` is able to successfully serve location requests. To this end, we considered the triple of parameters $(\alpha, \beta, \gamma) = (0.1, 0.75, 0.1)$ and performed location requests for increasingly rare objects. We chose these low values for `link` parameters because it aims at avoiding that the search phase (and as a byproduct the resource exchange) indiscriminately jumps on the different network locations thus possibly increasing transit fees.

Figure 7 shows results that `link` yields values of p_{hit} that are order of magnitudes higher than resource popularity even for rather scarce object diffusion. Of course, the scarcer the resource the higher the number of CNS to contact

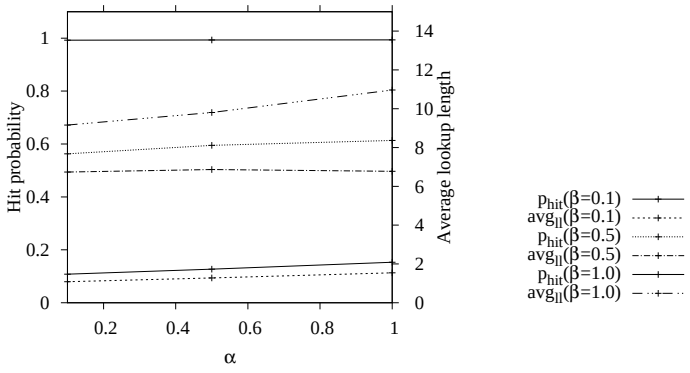

Fig. 6. Values of p_{hit} and avg_{ll} during the lookup as function of pair (α, β) for $\gamma = 0$.

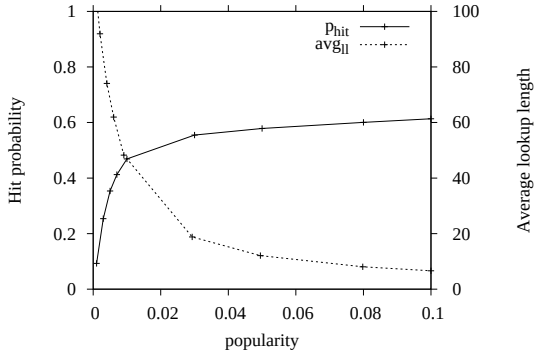

Fig. 7. Values of p_{hit} and avg_{ll} during the lookup as a function of popularity for parameters $(\alpha, \beta, \gamma) = (0.1, 0.75, 0.1)$.

before finding one that owns a copy. Indeed, avg_{ll} values increase as resource popularity decreases although the average lookup length for the scarcer resource is only 0.2% of the size of the AS snapshot we use for experiments.

As a final remark, please note that although the analysis we presented does not account for the dynamic evolution of the resource popularity (i.e., we are assuming here that the resource popularity does not change during the lookup phase), the insight it provides can be used by the CNSs to explore a wide set of parameters vs. the resource popularity.

In particular, performance can be tuned by letting each CNS modulate the costs of the lookup phase in terms of number of explored CNSs (and hence of the distance in terms of AS hops). In other words, the lookup algorithm can modulate the CNS's network awareness by tuning parameters (α, β, γ) to balance costs and expected p_{hit} since each CNS is aware of its connectivity relations and of the transit costs related with these relations.

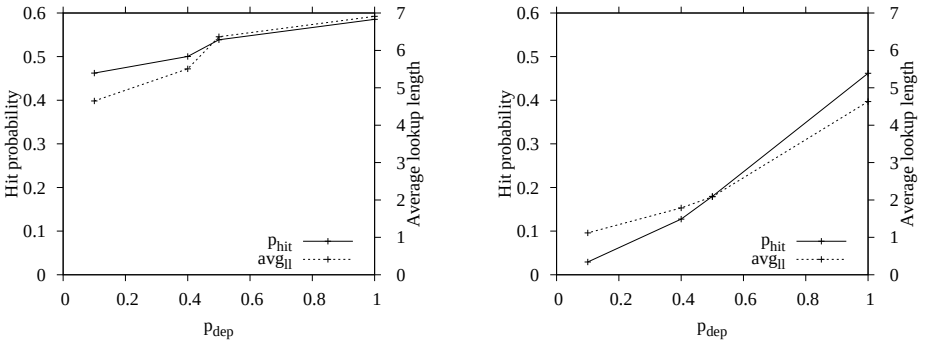

Fig. 8. Values of p_{hit} and avg_{ll} during the lookup as a function of p_{dep} for parameters $(\alpha, \beta, \gamma) = (0.1, 0.75, 0.1)$ and popularity 0.1 (left). The case with $(\alpha, \beta, \gamma) = (0.1, 0.75, 0.1)$ and same resource popularity for un-cooperating Tier-1 ASes (right).

3.3 Sensitivity to Deployment of CNS

Here we evaluate the performance of `link` as a function of how widespread CNS are in the entire network.

To this end, Figure 8 (left plot) shows results when Tier-1 ASes deploy a CNS with a certain probability p_{dep} for $(\alpha, \beta, \gamma) = (0.1, 0.75, 0.1)$ and resource popularity equal to 0.1.

It can be noted the contribution of Tier-1 ASes to the performance of `link` is not so high. Indeed, when Tier-1 ASes do not cooperate during the lookup we obtain $p_{hit} = 0.461884$ that is moderately less than the highest possible value, i.e., 0.58548. This can be explained by noting that Tier-2 ASes are generally well connected with many peers and many customers. This means that `link` can easily give up Tier-1 ASes and still be able to provide very good chances to successfully locate objects.

We further consider the case where no Tier-1 AS deploys a CNS and both Tier-2 and Tier-3 cooperate with probability p_{dep}. Results are summarized in Figure 8 (right plot); it can be noted that at least 40% of ASes should deploy a CNS to obtain a hit probability value that is greater than the resource popularity.

The last set of results characterizes a system where Tier-1 ASes do not cooperate while all Tier-2 ASes do. We consider varying levels of cooperation of Tier-3 ASes (the majority of ASes in the CAIDA snapshot) modeled by the adoption probability p_{dep}.

Results are reported in Figure 9; they show that `link` performance are only slightly degraded when only 10% of Tier-3 ASes cooperate in the search process by adopting a CNS. This means that adoption of a CNS can progressively start by incentivizing Tier-2 ASes to participate in the lookup framework.

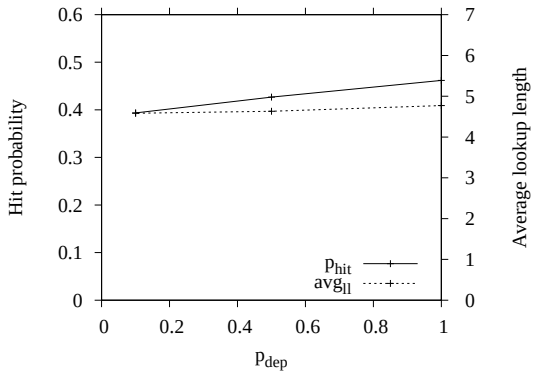

Fig. 9. Values of p_{hit} and avg_{ll} during the lookup for cooperating Tier-2 ASes, as a function of cooperation of Tier-3 ASes (probability p_{dep}) for parameters $(\alpha, \beta, \gamma) = (0.1, 0.75, 0.1)$ and popularity 0.1.

4 Further Developments

This section presents some improvements and features that could be explored and included in CNS Service.

Discovery Improvements.

1. To improve queries hit and limit messages, CNS can put in a cache the result of a successful queries lookup giving positive results not in the current AS. The positive effect of caches applied to all the CNS databases can leverage the number of message exchanges between CNSes;
2. To reduce traffic and flooding attacks, each CNS can limit the number of `link` packets arriving from an AS-customer, AS-provider and AS-peer; their number can be fixed on a AS-to-AS basis;
3. To improve liveness, a liveness politics can be implemented (see the Kademlia's bucket-table ordering republication [8]). Each publication in an authoritative CNS can have a lifespan: after the end of the lifespan, either the publisher re-publish the content in the CNS, or the record is simply dropped out from the CNS;
4. To limit the research space, a TTL can be introduced in `link` messages; TTL allows to limits the lifetime of lookup messages. A TTL counter attached to each link message allows to flush messages whose counter has elapsed;
5. To improve participation, incentives to locally republish contents retrieved abroad can be introduced: republication can be a simple pointer to another CNS. A tit-for-tat strategy could be installed between clients (looking for contents) and purveyors (distributing the contents) were the CNS should play a special role being in the middle of the above two actors;

6. To improve load distribution, CNS can perform load distribution among replicated copies of a single content. If CNS tables map a hypername into a lists of IP, then the CNS can respond with the entire list of purveyors, or it can rotate the ordering of the addresses within each reply. As such, IP rotation performed by CNS can distribute among multiple purveyors;
7. To improve the discovery success rate and focus the discovery search, each CNS can dynamically refine their α, β and γ flooding parameters by combining with the success probability of a given tag in the previous queries.

Content Aggregation in CNS. The data quality can be compromised by many factors, including data entry errors (*"OpneOffice"* instead of *"OpenOffice"*), missing integrity constraints (*"eat before December 12018"*), multiple convention (*"1st, rue Prés. Wilson, Antibes"*, versus *"1, rue du Président Wilson, Antibes"*), optional arguments ("+33(0)678123456" versus "0033678123456"), see [4] for a survey of data deduplication techniques. For a simple intuition, let the following hypername:

```
HN1 = fing_cont:hosts1:tags1:cont_name1
```

be published in some CNS and let

```
HN2 = fing_cont:hosts2:tags2:cont_name2
```

be retrieved by a `link` query: `HN1` and `HN2` differ in content names and in all logical attributes but the digital signature of the content `fing_cont`, which is the same. Because the digital signature is the same, the two hypernames should be merged into a single one. More generally, each time a purveyor publishes an immutable content with a given `HN2`, or a query return a list of purveyors, the authoritative CNS should verify that the same content is not already published with a similar but equationally different `HN1`[8] and, when it is the case, merge the two entries. Content aggregation should rewrite the previous two entries and substitute with the following ones:

```
HN1 = link to HN3
HN2 = link to HN3
HN3 = fing_cont:hosts1,hosts2:tags1,tags2:cont_name1|cont_name2
```

where the symbol "`,`" denotes list concatenation and the symbol "`|`" denotes an "or" operator that allow to match both content names in pattern matching.

Mobility. Since traffic from wireless and mobile devices has exceeded traffic from wired devices, most contents are requested and delivered by both wireless and mobile devices. It is well known that wireless and mobile devices may easily switch networks, changing their IP address and thus introducing new communication modalities based on intermittent and, possibly, opportunistic connectivity [12]. The CNS Service Discovery should be able to deal with mobility in case the owner/purveyor is a mobile host.

[8] E.g. synchronizing mail or telephone contact across multiple google accounts.

Nomadism. When a mobile node wants to publish a content, two cases can happen according to the (im)mutability of the content:

- *immutable:* (most common of the two). The authoritative CNS related to the mobile Internet provider accept the publication of an immutable content by a mobile user with the proviso of *i)* recording the identity of the user, via e.g. the MAC address of the mobile device (or another identifier of the mobile node), and *ii)* asking to the mobile user to re-publish the content more frequently than a fixed device, and *iii)* possibly blacklisting a mobile device that appear and disappear too fast or too often.
- *mutable:* it deal with the possibility to keep an identity also in case the user is navigating through different mobile networks. The authoritative CNS related to the mobile ISP could accept the publication of a mutable content if and only if the logical attribute `fing_princ` is present and the logical attribute `hosts` contains only one symbolic name or only one IP.

Security. Until now, a few significant DNS attacks has corrupted the DNS service: this is because *i)* DNS Servers are machines managed and protected by system administrators, *ii)* the DNS protocol pushes lookup always "below" the hierarchical database, minimizing the "uphill ascents", and *iii)* of making use of cache techniques. We think that the above arguments could be applied also the CNS Service because the relatively fixed number of CNS servers (∼70K) could be managed by AS system administrators, and `link` always pushes the location process first downhill the customer-CNS distributed database, and, only in case of failure, uphill through a peer-CNS or a provider-CNS. Nevertheless, the CNS Discovery Service is not vaccinated by either DDoS bandwidth-flooding attack, or man in the middle attack, or poisoning attack, or spoofing an IP of a node below an authoritative CNS.

Acknowledgments

The work has been partially supported by the HOME (Hierarchical Open Manufacturing Europe) project, supported by the Regione Piemonte, Italia (framework program POR FESR 14/20).

References

1. Bari, M.F., et al.: A survey of naming and routing in information-centric networks. Communications Magazine, IEEE **50**(12), 44–53 (2012)
2. CAIDA: Center for Applied Internet Data Analysis: AS relationship. `http://www.caida.org/data/as-relationships/` (2016)
3. Chand, R., Cosnard, M., Liquori, L.: Powerful Resource Discovery for Arigatoni Overlay Network. Future Generation Computer Systems **24**(1), 31–48 (2008)
4. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. on Knowl. and Data Eng. **19**(1), 1–16 (2007)

5. Gao, L.: On inferring autonomous system relationships in the internet. IEEE/ACM Trans. Netw. **9**(6), 733–745 (2001)
6. Jacobson, V., et al.: Networking named content. In: Proc. of CoNEXT. ACM (2009)
7. Koponen, T., et al.: A data-oriented (and beyond) network architecture. SIGCOMM Comput. Commun. Rev. **37**(4) (2007)
8. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: Proc. of IPTPS (2002)
9. Mockapetris, P.: Domain names - concepts and facilities. `https://tools.ietf.org/html/rfc882` (1983), RCF 883, updated 973, 1034, 1035
10. Rekhter, Y., Li, T.: A Border Gateway Protocol 4 (BGP-4). `https://tools.ietf.org/html/rfc4271` (1995), RCF 4271, obsoletes 1654, 1267, 1163, 1105
11. Shang, W., et al: Named data networking of things. In: Proc. of IEEE IoTDI (2016)
12. Xylomenos, G., et al: A Survey of Information-Centric Networking Research. IEEE Communications Surveys & Tutorials **16**(2) (2014)
13. Zhang, L., et al: Named data networking. Computer Communication Review **44**(3), 66–73 (2014)