Object Centric Networking, from Database to Networked Objects

Hidden due to double blind review
Affiliation placeholder
email placeholder

***Abstract*—Data is key and is the raw material to produce services of the future. An enormous set of data is going to be generated by various sources such as cloud, objects, sensors, and humans. The issue is therefore to trade these data to feed the relevant applications and provide opportunities to business creation or public services. We argue for the development of a service that will provide a seamless mediation between applications/consumers and data/producers. Named Data Networking (NDN) purpose seems to fit this need. We applaud the idea to identify an object with a name.**

Nevertheless, such a matching is solely based on the naming and thus do not consider the nature and structure of the content. In this paper, we demonstrate the benefit of a network architecture where contents are represented by attributes and methods, namely managing multi-dimensional objects interconnected with semantic relationships. Our contribution is threefold.

First, we illustrate the value of bringing this information in the network. Second, we present a routing protocol allowing to transparently match the data produced to consumers' interests. Third, we show how relational algebra can improve data retrieval in this paradigm. We conclude by presenting implementation of our solution.

Introduction

The goal of services is to bring data from equipments toward its consumers. However, IP networks are by nature host-centric, whereas services are by essence data-centric.

This observation motivate the need to design a mediator able to match applications/consumers interests with the data provided by the producers of the network. Named Data Networking is a first attempt which seems to fit this need. By identifying a content using a name and routing interest according to this name, NDN is able to retrieve raw data using pub/sub mechanism while abstracting the content location. It then allows to have in one hand users interests and in the other the data produced by each equipments and perform a transparent matching.

Nevertheless, the key point to build new services is the objects produced by each equipments and the semantic links connecting those objects. Naming is not sufficient to accurately describe the nature of the contents and such links. To do so, we have to characterize more accurately the content. In Object Centric Network (OCN), we propose to represent content using an object model. By offering this additional knowledge in the network, one could express exactly high-level and accurate interests to the network and get the corresponding data. Hence, OCN transforms the network into a market place able to match the interests of the data consumers (user applications, ...) with the data traded by the providers (clouds, databases, and more generally all the existing services offered by the network). This paradigm aims at offering to new services a context were they can act both as data consumers and data providers. The way the data is traded reminds how IP routes are exchanged between Autonomous Systems in IP networks.

In Section I, we present how to model objects in OCN and the benefits of bringing this additional knowledge to the network. In Section II, we detail how objects are shared and exchanged in OCN and clarify the hypotheses needed to have a consistent distributed ecosystem of data. In Section III, we present a routing protocol over OCN allowing to match complex consumers' interests with the corresponding data. We demonstrate how database background can help to optimize routing table size in OCN and can optimize the data retrieval. In Section IV, we present a broker of monitoring data fed by several authorities built above the OCN paradigm. In Section V, we discuss our proposal and the remaining open questions. We conclude the paper by presenting the related work.

I. Objects in OCN

A. Naming objects and their attributes

Data produced is the raw material to build new services, and thus data consumers need to learn what is produced by each neighboring data provider. By representing each object instance by a name, a NDN network [1] is unaware of the nature of the data it transports. A workaround in NDN consists in encoding some information in the naming assigned to objects according to an arbitrary hierarchy.

[2] points out the difficulty to apply a hierarchical aggregation in ICN at the Internet scale with today's technology. Moreover such a hierarchy cannot explicit the links connecting objects semantically related, or even detect whether two contents are of same nature. To build new services, naming is not sufficient, since the network do not offer primitive allowing to gather content of same nature or combining content sharing semantic links.

That is why in OCN, we propose to represent contents using an object model, which is the natural approach adopted in Object Oriented Programming. Representing data in a network has already been proposed in Pub/sub context [3], in P2P overlays [4] and for sensor networks [5]. Building new services requires on the one hand, that the network provides transparent access to the data, and on the other hand a high level query language allowing to retrieve and combine this

```
class Movie {
    String title;
    Country country;
    Int    year;

    KEY(title);
}

class Country {
    String  name;
    Int     population;

    KEY(name);
}
```

Fig. 1. Two semantically related objects.

data [6], [7]. The OCN paradigm try to offer this kind of mediation.

In OCN, *objects* are defined according to a class name (*type*) and by a set of named (and typed) attributes, as shown in Figure 1. An object is a template describing a set of *instances* (contents).

Typing explicits the semantic relationships connecting the different objects. The underlying idea is to offer to new services an interface over the network allowing to characterize the objects of interest, their relevant attributes, and how to combine them. In other words, OCN aims at offering the same level of abstraction as an Oriented Object Programming Language over the network by bringing physical and logical independence.

Such a paradigm opens a new field of research. We propose in this paper to explore how such a network could be designed. So far we can already list several challenges:

1) *Access policies:* OCN is targeted to offer a market place to different actors trading data. The access to the data may be restricted due to privacy, rental, trust, or grants considerations. OCN should only match user interest to contents respecting these access policies. This challenge will be developed in Section II.
2) *Correctness:*
   a) *Global referential:* To provide a consistent ecosystem of distributed data, all the authorities must provide data, which is not contradictory and built in respect common convention. It includes exposing a consistent typing and compatible data representations. Since semantically linked objects may be managed by different authorities, all of them should adopt a same way to refer to a given instance object. If an instance can be identified and if it is provided by several data providers, each of them should not expose contradictory values. This need is explained in Section I-C).
   b) *Completeness:* For a given input query, an OCN network should fetch all the matching data (in respect of access policies).
3) *Managing data heterogeneity:* In practice, several data providers may offer different subset of attributes of a same object. OCN should be able to manage this heterogeneity. This challenge is tackled in Section II-D.
4) *Robustness:*
   a) *Scalability:* We expect a large number of objects in OCN, and hence require to summarize efficiently the objects surrounding an OCN node. How to represent concisely this ecosystem of data? We could describe very accurately all this ecosystem, but the overhead induced by such a level of details would be to the detriment of scalability. Reciprocally, if this description is too inaccurate, a large part of the network may be flooded to find the requested data.
   b) *Dynamics:* Since contents may appear and network failure may occur anywhere, OCN should offer a routing resilient to such dynamics.

B. Benefits

Thanks to this additional knowledge, the network can handle higher level queries. First, such a network allows client consumer to express more accurately its interest. It can take advantage of this to optimize the data retrieval to only fetch the relevant attributes of the queried object and focus on the relevant data providers. Second, such a network can take advantage of the semantic it learns to transparently aggregate objects semantically related. Third, this model allows to detect whether the interest of a given client is included in the interest of another client. In a pub/sub context, we can detect such inclusions and thus offer a native generalized multicast.

C. Linking objects through typing and keying

Using typed attributes explicits the semantic relationship connecting the pointer and the pointed objects. Let be a an instance which *refers* to an instance b. The instance a must store an identifier allowing to find the corresponding pointed instance b. If those two objects are managed by two different authorities, they must then adopt a common convention. a will basically store an identifier understandable by the authority providing b.

We call *key* a minimal subset of attributes of an object allowing to identify any instance of this object. For example `name` is a key of the `Country` object. In general, an object may have one or several keys. The benefits of keying object are twofold. First, keys can be used to encode reference between two related objects without ambiguity. For example, a Movie provider can refer to a Country using the corresponding `name`. Second, a node in the network can take advantage of the keys it collect for a given interest to detect whether the corresponding instances are distinct and eventually remove the duplicates.

However, object typing and keying, attribute naming clearly highlights the need of a common referential to federate objects over a set of authorities. Indeed, if objects are not consistently typed, identified or referenced, OCN may perform wrong combinations of objects (if keying or typing are wrong), miss combinations (if keying is incomplete), return incomplete results (if objects are not consistently named) or wrong results (if data provider do not conform to the same units).

II. Sharing objects in OCN

A. Trading objects according to routing policies

Sharing objects in OCN consists for a data provider in announcing the structure of the object(s) it serves, as depicted

in Figure 1. A provider announcing an object to a consumer implicitly trusts this consumer. Reciprocally, a consumer accepting an announcement trusts the corresponding data.

For sake of simplicity, we assume in this section that all the data producers conforms to a *common referential* and expose objects having the same keys and structure. We will discuss in Section II-B and in Section II-D about some workarounds to relax this assumption.

Our proposal is clearly inspired of BGP [8], the current inter-domain routing protocol used in the Internet. The Internet is made of thousands of interconnected domains, called ASes, containing routers exchanging IP routes. BGP relies on destination announcement and withdrawal. Announcements are ruled by the routing policy configured on each router. BGP routing policies hence encode in the network how IP resources are traded. Organizing a network in ASes offer several advantages. First, each authority controls the access to the data it provides and from which neighbors retrieve data. Inter-domain paths result of a chain of confidence between neighboring ASes. Second, splitting the network in AS allows each authority to reorganize its infrastructure without impacting the rest of the network.

We propose to adopt the same design in OCN. In OCN, ASes correspond to a service (or a set of services) ruled by a same authority. Routers exchange objects with its neighbors according to a routing policy encoding how objects (destinations) are exchanged between partners ASes. As in BGP, router can only re-announce a subset of the destinations it learns. OCN would be hence become a support to trade objects between services and should built around mutual interests.

Using a "per-neighbor" routing protocol seems to be a better approach than a link-state protocol, which maintains a map of the whole network. Indeed, we expect to have a large number of services and authorities, all over the network. By design, we preserve routers to maintain such a map. Moreover, we can keep private the routing policy of each AS.

B. The OCN destination space

Partial connectivity: A major difference with IP networks is that ASes do not try to get the reachability to all the destination objects. Rather OCN nodes are only interested in a small fraction of the destination objects (those involved in their services) and will establish relationship with AS interested in or providing those objects. By nature, the OCN meshing reflects the semantic relationships connecting the different services together. As a consequence, the scope of objects in OCN is restricted to a specific part of the network. Hence each partner AS only obeys to a referential shared with its direct neighbors, which avoid to have all the ASes conforming to a same arbitrary referential. This distinguishes OCN from approaches relying on a same global referential as those proposed in P2P, where all the objects are gathered in a DHT.

Open world assumption: In IP, an AS announcing a prefix is supposed to serve the whole prefix. This can be guaranteed by the network since IP address allocation is controlled. This is not the case in OCN; contents are by essence volatile and may appear anywhere and created anytime. An AS announcing an object only expose a template describing roughly the data it offers. However, it do not claim to produce all the possible corresponding instances (and cannot claim this if it is not authoritative over this object). As a consequence, OCN routers necessarily *over-announce* the data they offer.

C. Implications due to over-announcement

The over-announcement performed by each router (see SectionII-B) directly impacts regarding the routing and the forwarding in OCN.

Let us consider a router learning announces toward a same given object from several peers. Without additional information, this router is unable to determine whether those announcements correspond to overlapping or distinct instances, and so all of them are potentially relevant. In terms of routing, a router cannot select a single route toward a given destination since all of the candidate routes may lead to different content and are thus relevant. In terms of forwarding, guaranteeing completeness for query related to this object implies that this router query all its relevant peers. If some peers provide the same instances, then they will be retrieved several time. More generally, if several paths exists between the data consumer and the equipment producing the data of interest, this data of interest can flow along each OCN paths established between them, leading to an overhead of network resources consumption.

A possible workaround consists in listing AS path that must be traversed to reach the destination, as done in BGP. In practice, an AS simply has to prepend its own global AS identifier in the AS path attribute of the announcement. The benefits of putting AS path also in OCN are twofold. First, the AS path can be used to prevent announcements to flow along a cycle of ASes. Second, if we assume that an AS will never filter instances regarding the object it re-announces, the announcement related to a same given object and issued by a same *origin* AS can now been compared, while preserving completeness. If so, an AS could only select a subset of routes (or even a single best route) toward each (object, origin) pair according to a decision process. If all the routers route objects according to a single best path, then by construction, they only fetch the data along the reverse path. We assume that the retrieval path is unique in the rest of the paper without loss of generality since a node could discard duplicates using keys. Hence a consumer interested on a object contact the different data owners through a structure of *tree*, where each node of the tree correspond to ASes forwarding its interest to a data owner.

Any decision process could be considered since it does not break our assumption. Choosing the best possible decision process remains an open question. It highly depends on the metrics we try to optimize and the routing properties we desire (convergence, stability, determinism, etc.) and which are not guaranteed by the BGP decision process.

Note that bringing the origin in the destination is not contradictory with offering a location-independent interface over the network. To query a given object, a router simply has to route its query according a set of routes covering all the corresponding (object, origin) pairs it learns. Moreover, adding origin in the destination brings two benefits. In one hand,

the data consumer can discard (object, origin) announcements it does not trust. In the other hand, a data consumer only interested in the data produced by a specific origin is able to route consequently its query.

D. Sharing heterogeneous objects

Same objects with different attributes: Depending on their specialization, data providers will only provide a subset of the attributes corresponding to a given object. However the OCN model remains tractable while key needed by a consumer are supported by its direct providers. Keying indeed allows for each returned instance to merge the pieces returned by each provider. This relaxes the *common referential* assumption. By allowing this heterogeneity, we also allow an AS to re-announce an object with only a subset of its attributes depending on its consumers. It typically concerns private attributes or for sale attributes.

Specialized objects: In Object Programming, inheritance allows to specialize a child class using a parent class. The attributes of the child object C includes those of the parent object P. If we omit the name of the parent and child classes, this is a particular case of the previous paragraph. If an OCN router learns such an inheritance, it can forward queries related to P irrespectively to neighbors offering C or P. The router can also only re-announce C to neighbors interested in C and P. For sake of simplicity, we won't consider inheritance between objects in the rest of the paper.

III. Matching interests and contents

A. Overview

In OCN, we offer a query language to data consumers allowing them to query objects according to their name and properties, while abstracting their location in the network. The goal of OCN consists in matching interest (formulated through consumers' queries) with the corresponding data (exposed by providers' announces).

The OCN query protocol is build around three kind of packets:

- A *query packet* transports the interest of a data consumer toward a neighboring data provider.
- A *record packet* transports an instance returned by a data producer toward a data consumer. A record is basically a dictionary key/value where the key corresponds to the name of an attribute.
- An *error packet* transports errors (typically returned by a data provider to a consumer, for instance due to a timeout). See III-F for further details.

The way a consumer expresses to its neighbor directly depends on the objects exposed by its neighbors (see Section II). Since those announcements are in general heterogeneous (see Section II-D), each neighbor might be queried differently for a given interest. We call *query plan* this decision (see Section III-C). Once computed, the consumer can craft dedicated query packets for each relevant neighbor and wait for the resulting record packets. Each of the record packets are aggregated and combined according to the query plan to craft the final result.

```
SELECT  title, country
  FROM  Movie
  WHERE year == 2014
```

Fig. 2. A simple query in OCN

When a data provider v handles a query packet, it memorizes which consumer u has issued the query in its PIT (Pending Interest Table). If it owns (or caches) the requested data, it simply returns the record packets corresponding to the requested instances. It then acts as if it was a consumer. Therefore, it also computes a query plan to discover whether its own neighbors propose data of interest. The records resulting of the query plan computed by v are send back to u.

When a data provider v handles a query packet, it memorizes which consumer u has issued the query in its PIT (Pending Interest Table). If v owns (or caches) the requested data, it simply returns the record packets corresponding to the requested instances to u. Otherwise, v then acts as if it was a consumer. Therefore, it also computes a query plan to discover whether its own neighbors propose data of interest. The query is then propagated from neighbor to neighbors that claim to have the requested data. The records resulting of the query plan computed by v are in the end send back to u.

As explained in see Section II-C, query packets are in practice forwarded along a tree of ASes, where the root is the initial data consumer and the leaves are data providers of interest. The other ASes of the tree act as brokers between their consumer and their provider(s). Each traversed router encodes a part of the overall query plan. The resulting data is transported along the reverse tree (from the leavers to the root).

Not clear... A consumer is interested only in a subset of fields or specific object instances could retrieve the whole set of complete and instance and perform pruning at the edge. However, such an approach is costly in terms of network resources. In Section III-D, we show how relational algebra can be used to optimize the distributed query plan.

B. Querying an object in OCN

We propose to adopt for OCN a query language inspired from databases. A query specifies which object is queried, what are the attributes of interest, and eventually conditions which properties must be verified by the requested instances (*records*). This namely corresponds to the `FROM`, `SELECT`, and `WHERE` clauses in the example depicted in Figure 6.

As we can see, the query is object-centric and thus abstracts the data owner and the data location(s). So far, computing a query plan simply consists in enumerating which neighbors offer the `Movie` object with at least the `title`, `country` and `year` attributes.

Indeed, only the providers offering the fields involved in the `SELECT` clause can return instances having all the requested attribute. And only the providers having attributes involved in the `WHERE` clause can guarantee they do not return false positives. A query is *feasible* for a data producer if and only if it provides the object involved in the `FROM` clause and if it knows all the fields involved in the `SELECT` and the `WHERE`.

```
SELECT  title, country
  FROM  Movie
  WHERE country == "USA"

SELECT  title, country
  FROM  Movie
  WHERE director.nationality == "USA"
```

Fig. 3. A simple query in OCN

Since a consumer cannot guess how many records will be return by each of its providers, in our proposal, producers mark the last record they return by enabling a dedicated flag called `LAST_RECORD`. Once all the producers have returned such a record packet, the consumer knows it has collected all the relevant data, and then can mark the last record resulting from its local query plan. Note that for a given feasible query, a producer might have no corresponding instances. If so, it simply returns an empty packet record having its `LAST_RECORD` enabled.

Such a query only relies on a single object, and our system only offers *filtering* capabilities (based on the object attributes) and a transparent *aggregation* of the matching instances disseminated over the network. This corresponds to the Selection, Projection, and Union operation in databases. This query language is comparable to the one proposed by CBN [9].

C. Computing a query plan over several shared objects

OCN routers learn the semantic relationships connecting objects to each others. Therefore they can exploit this additional knowledge to *combine* objects of different nature.

The key idea consist in detecting whether two different objects are connected through a unique semantic path, where each step of the path correspond to an object. For example, "a american Movie" is "a Movie such that its Producer is based in a city in USA". We cannot expect that each data producer would express itsr content using the same object decomposition. However we clearly see in this example that there is no ambiguity and that those both formulations clearly identify the same contents. In this case, the first formulation is a *shortcut* of the second formulation. It does not mean that this is the only way to connect two given objects. For instance a Movie can be matched with the nationality of its director. However, we would request for "a Movie having an american director", and clearly explicit that the City and the Movie object are matched through a Director object. Since OCN query language should allow to fetch this second group of content, our query language requires to characterize distinctly those two groups of instances. As depicted in Figure 3, we propose to explicit intermediary objects which are not along a semantic shortcut.

Enumerating objects involved along a shortcut (if any) is not mandatory. But it clearly highlights that providers may announce an object, which is in practice a *view* over several objects and induced by implicit semantic shortcuts.

The first step consists in finding a common referential to represent the object announced by each neighboring router. We assume all the object are correctly keyed, e.g. that a key cannot be reduced to a smaller subset of attributes, and can be used to distinguish any instance of the object. An object can be represented equivalently with a set of *functional dependencies* associating a key with its corresponding attributes. Since in OCN, we require to keep track of which objects and attributes are provided by which peer, OCN functional dependencies map a (key, neighbor) pair (we call this pair *determinant*) with the appropriate set of attributes.

The second step consists in decomposing each object announced by each peer (views) into a common set of (normalized) tables, resulting from a set of gathered functional dependencies. **unclear, il faudrait que jordan m'aide clarifier l'algo car je ne suis pas capable de l'expliquer clairement**

1) Split each input table into functional dependencies.
2) Find a minimal cover set of functional dependencies according to a classic 3nf algorithm
3) Organize functional dependencies according to a object/neighbor hierarchy.
4) Construct each resulting normalized table.

Each announced table is hence translated as a view joining a subset of normalized tables. This abstraction offers a level of indirection allowing to find a common referential among the objects published by the neighboring routers. Those two first steps only depend on object announcements and are thus a part of the routing plan. Therefore normalized schema and the mapping with all the corresponding input objects forms the routing information base (RIB) of an OCN router. This RIB must be updated upon object advertisement, update, or withdrawal. Note that at least, a query still only specifies a single root object in `FROM`. All the implicitly involved objects remains transparent to the consumer. In other words and by contrast with databases, a consumer never specifies the chain of `UNION` and `JOIN` is involved in the `FROM` clause, providing a form of logical independence.

TODO: donner un petit exemple de table non normalises TODO: dcrire la structure du graphe 3nf

The third step consist in finding all the relevant normalized tables according to an input query.

1) Starting from the object involved in the `FROM` clause find all the attributes of interest (involved in the `SELECT` and `WHERE` clauses). If a intermediate attribute is specified (e.g. `producer.nationality`), the algorithm will be re-run starting the intermediate object. Note that since no semantic ambiguity can exists, each run of this step return a tree covering all the attribute of interests. Otherwise, the query is unfeasible.
2) Translate each tree into a neighbor-dependant query plan. The `FROM` clause now specifies which Unions and Joins are performed over the normalized tables.
3) Translate the query plan the normalized schema to fit with the table announced by each peer. The query plan is basically a tree where leaves plan corresponds to the relevant objects the neighbors. The other nodes corresponds either to `JOIN` operators (joining object of different nature and semantically related), and `UNION` operators gathering objects of same type. **TODO jordan: subquery**

4) Place the SELECT and WHERE operator on top of the query plan.

TODO: faire un dessin sur le qp correspondant, idalement distribu sur plusieurs noeuds The step depends on the input query, and is thus a part of the forwarding plane. It basically prepare the pipes which will retrieve the data, and the operations which will transform the data along the pipes.

D. Optimizing the forwarding plan thanks to relational algebra

The fourth and last step is not mandatory to have a system returning correct result, but helps to preserve network resources. It consist in optimizing the query plan to fetch a minimal amount of data by taking advantage of well-known optimizations related to database background [10]. The underlying idea consist in discarding irrelevant data as soon as possible. To do so, we try to by push as close as possible the Selection and Projection operators close to the leaves of the query plan. If a Selection or a Projection operator reach a leaves, this means that operation can be pushed to the neighbor. It will impact consequently the SELECT and WHERE clause of the query packet send to this neighbor.

TODO: montrer un le qp optimis correspodant

a) OCN meets CBN:: Suppose that data providers can provides clauses (based on the object's attributes) describing which instances it provide. Then will optimizing the query plan, we could detect whether a neighbor is a irrelevant candidate for specific input queries. For example, a data provider specialized in american movies cannot serve queries related to french movies. This is basically the idea proposed in CBN networks [9]. By providing such an additional knowledge we could prune some irrelevant neighbors.

E. Optimizing the PIT for pub/sub queries

OCN can support a pub/sub mechanism and offer natively a generalization of multicast.

We consider a data provider handling two input queries q_1 and q_2, such as the:

- The set of instances characterized by q_1 is included in (or equal to) the set of instances characterized by q_2. In other words, the q_1's WHERE clause it more (or equally) restrictive than the q_1's one.
- The set of attributes requested by the q_1's SELECT clause is included in the set of attributes corresponding to the q_2's SELECT clause.

If so, q_1 is *included* in q_2 ($q_1 \subseteq q_2$). Indeed, all the instances resulting to q_1 can be computed using the instances resulting to q_2. To merge those two interests, the router simply have to build the q_2's query plan. The corresponding root node have two roles. First, by construction, it collects and therefore can be used to produced the packet records for each neighboring consumers having subscribed to q_1. Second, since q_2's query plan is equivalent to perform q_1's query plan plus appending a Projection and Selection operators corresponding to q_2, q_1's root node can be derived to populate q_2's query plan. Hence, the router can take advantage of the semantic it perceives of its consumer to decrease the amount of data it fetches.

If $q_1 = q_2$, offering such a subscription aggregation is equivalent to offer a native support of multicast. Since in OCN, a router is even able to detect whether $q_1 \subseteq q_2$ each branches of the tree connecting the router from the final consumers, the content is not anymore the same.

Detecting such inclusions simply consists in maintaining in memory a lattice of pending queries built according to $\subseteq$ operator. Each node of the lattice points to its corresponding pending query and query plan. This structure allows first to easily detect whether a incoming query is included in a pending query. If so it is straightforward to merge the both resulting query plans.

- fw: - inheritance: ping and traceroute are measurements - clustering ludo - 1 + 1 = 3 - combining data (join) - optimizing queries - capabilities

F. Managing errors in OCN

peut-être pas fondamental OCN network will face to many causes of failure and thus must be resilient to errors. We already pointed out that an unfeasible query may appear.

- *Credentials:* the access to a given data may be restricted to a set of authorized user. By transporting in the query packet the user credentials, a data provider can recognize whether a consumer is authorized to access a given data and otherwise, return an error packet.
- *Timeout:* if a router perform run a query which takes too much time, it can interrupt its query. If so, it can notify its eventual consumers that this query has been interrupted using an error packet.
- *Grants:* suppose that announcement now express which attributes are read-only or read-write. A consumer issuing a forbidden query will collect an ERROR packet.

IV. A PROTOTYPE FOR OCN

This section presents how we have built a broker relying on the OCN paradigm and offering monitoring data produced by different authorities.

Architecture: Figure 4 depicts the ecosystem of producers and consumers involved in this architecture:

- *Maxmind* offers an API able to return for a given input IP its geolocation, the corresponding city and the corresponding country.
- *TeamCymru* exposes information mapping an IP with a corresponding BGP announcement such as the corresponding prefix and ASN of its origin AS.
- The *controller* federates a collection of OCN nodes (corresponding to PlanetLab nodes in our use-case) performing periodical and on-demand network measurements.

In Figure 4, ellipses stands for equipment supporting OCN, whereas rectangle represents source of data wrapped in dedicated translators making them OCN-compliant. These translators allow the OCN ecosystem to inter-operate with the legacy services.

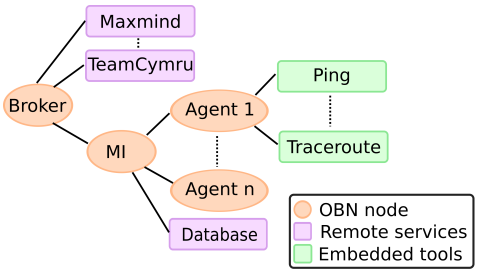

Fig. 4. A OCN ecosystem dedicated to federate network measurements.

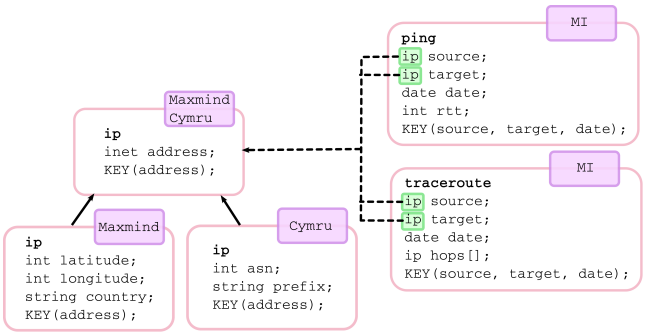

Fig. 5. The broker's RIB contains the schema of objects it perceives.

```
SELECT  source, target, hops.asn
  FROM  traceroute
  WHERE timestamp == now

SELECT  source, rtt
  FROM  ping
  WHERE timestamp == now AND target == 8.8.8.8
```

Fig. 6. A simple query in OCN

Routing plan: Each measurement agent announces the objects corresponding to the network measurement tools it supports. Each of them can push periodical network measurements in a database which can be later be queried by the controller. The controller re-announces the `Ping` and `Traceroute` objects to the broker, and thus hide the measurement architecture complexity to the broker. The broker learns announcements issued by the controller. The announces representing the `Ip` object provided by the Maxmind and the TeamCymru translators are statically configured in its RIB. According to those announces, the broker computes the normalized shown in Figure 5. Solid arrows depicts the specifics view provided by each neighbors, whereas dashed arrow represents the semantic links deduced thanks to typing and achieved thanks to a common identifier (here, an IP address). In the similar manner, the controller stores a schema made of the `ping` and `traceroute` objects served by the agents and the database.

Extending the monitoring architecture: Note this architecture allow to plug new services and new measurement agents easily, as illustrated by dashed line in Figure 4. If the number of measurement agent become large, scalability issue may arise. We straightforward solution consist in design a hierarchy of controllers. Another nice approach would consists in offering a self-organized overlay of agent guaranteeing that the degree of each agent and of the controller remain small, as proposed in [11].

Forwarding plan: The user can visualize monitoring information through a web portal which queries the broker to retrieve the information to display. Depending on the timestamp (a given date or the `now` keyword) formulated in the input query, the controller can determine whether it corresponds to an on-demand or an archived measurement, and will consequently either query the database or either query the relevant set of agents. It simply requires that the announcements corresponding to the objects stored in the database specify that the timestamp attribute is different of `now`.

Expressiveness brought by OCN: A major benefit of OCN is its ability to hide the complexity induced by the underlying sources of data used by the queried node. Suppose that measurement agents are deployed over PlanetLab [12]. OCN allows the broker to issue high scale and world-wide measurements with very concise query, as demonstrated the two queries proposed in **??**:

- The first query returns a set of AS paths deduced from a set of live traceroute measurements, and could used to enrich public BGP datasets, actually only perform by a small set of BGP trace collectors.
- The second query performs delay measurement from each node toward the the 8.8.8.8 anycast IP and can be used to discover the underlying anycast architecture [13].

One may imagine various other world-wide and high scale measurement campaign to detect other phenomenon like black holes (by probing periodically from various node a given monitored IP address), censorship (by performing http request all around the world), impact of major outage like earthquakes (by joining network measurement and geolocation information).

V. Discussion

VI. Related work

Many Information Centric Networks (ICN) architectures have been proposed over the last decades. The goal always consists in providing an abstraction regarding the content location while minimizing the stretch to reach the data.

Named-centric approaches: TRIAD [14] is the first attempt to build an ICN. It consists in identifying contents using user-friendly and location independent names. The name-to-address resolution is performed by dedicated content routers rather than by a DNS server. DONA [15] extends the TRIAD approach and propose a alternative to DNS (which mapping a name with an host) by offering a resolver able to match a user-friendly flat name with a content. 4WARD/SAIL NetInf [16] offers a pub/sub architecture based on DONA-style names. The content are described according to information object (IO) describing the content according to different level of abstraction or point of views. Each IO may point to one or more data object (DO) containing a related content. The level of indirection offered by IO increase the expressiveness of the interest which may be formulated. COMET [17] aims at offering a mediation plane aware in the network status in order to offer content delivery with guaranteed QoS. NDN [1]

proposes a redesign of the routing architecture itself, where consumers request a content according to a name. This idea has provoked a wide interest in the research community.

Object-centric approaches: PSIRP/PURSUIT [18] propose to preconfigure rendezvous points in the network. In one hand data publisher characterize their content, in the other hands subscribers express their interest. A rendezpoint can achieve the data collected to its relevant consumers using multicast. In CBN/CBCB [9], content are identified by a set of key value pairs. A user interest consists in a clause of predicates over those attributes. By contrast with named-based approaches, the forwarding is no more based on longest prefix matching. CBN assume that one (or more) forwarding trees are pre-established over the network. Each arc describe thanks to a clause what content may be reached along the branch, allowing to only contact the relevant neighbors.

VII. Conclusion

In this paper, we have presented a new paradigm of Information Centric Network, called OBN, where content are described and manipulated as Objects. We have shown it is possible to offer a high level of query language over a network while abstracting the content locations. We have demonstrated how OBN paradigm could be used to trade data by doing a parallel over an ecosystem of actors providing data and services. We have proposed a routing protocol inspired of BGP allowing to encode complex business relationship, resilient to network and content dynamics. We've have proposed some technique to reduce the routing table size and optimize data retrieval. We've shown that a pub/sub architecture can be built above the OBN architecture and offer a native generalization of multicast. We have developed an open-source implementation of OBN. We've demonstrated through a real use case (a broker over monitoring data and tierce services provided by different authorities) the tractability of our proposal.

References

[1] V. Jacobson, M. Mosko, D. Smetters, and J. Garcia-Luna-Aceves, "Content-centric networking," *Whitepaper, Palo Alto Research Center*, pp. 2–4, 2007.

[2] A. Narayanan and D. Oran, "Ndn and ip routing can it scale," in *Proposed Information-Centric Networking Research Group (ICNRG), Side meeting at IETF-82, Taipei*, 2011.

[3] Y. Jin and R. Strom, "Relational subscription middleware for internet-scale publish-subscribe," in *Proceedings of the 2nd international workshop on Distributed event-based systems*. ACM, 2003, pp. 1–8.

[4] C. S. D. UC Berkeley, "Pier project." [Online]. Available: http://telegraph.cs.berkeley.edu/tinydb/

[5] TinyDB, "Tinydb." [Online]. Available: http://telegraph.cs.berkeley.edu/tinydb/

[6] J. Hellerstein, "Adaptative dataflow: A database/networking," 2000. [Online]. Available: http://db.cs.berkeley.edu/jmh/talks/DBNetsConverge.ppt

[7] S. Fdida and M. Diallo, "The network is a database," in *Proceedings of the 4th Asian Conference on Internet Engineering*. ACM, 2008, pp. 1–6.

[8] Y. Rekhter and T. Li, "A border gateway protocol 4 (bgp-4)," United States, 1995.

[9] A. Carzaniga and A. L. Wolf, "Forwarding in a content-based network," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 163–174.

[10] "Use of algebraic properties for query optimization." [Online]. Available: http://en.wikipedia.org/wiki/Relational_algebra#Use_of_algebraic_properties_for_query_optimization

[11] C. Peplin, "Massively distributed monitoring," 2011.

[12] 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. 3–12, 2003.

[13] "Name placeholder." [Online]. Available: URLplaceholder

[14] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet." in *USITS*, vol. 1, 2001, pp. 4–4.

[15] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 181–192, 2007.

[16] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, "Design considerations for a network of information," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, p. 66.

[17] G. García, A. Beben, F. J. Ramón, A. Maeso, I. Psaras, G. Pavlou, N. Wang, J. Sliwinski, S. Spirou, S. Soursos *et al.*, "Comet: Content mediator architecture for content-aware networks," in *Future Network & Mobile Summit (FutureNetw), 2011*. IEEE, 2011, pp. 1–8.

[18] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From psirp to pursuit," in *Broadband Communications, Networks, and Systems*. Springer, 2012, pp. 1–13.