OpenGeoBase: Information Centric Networking meets Spatial Database applications

Andrea Detti, Nicola Blefari Melazzi,, Michele Orru, Riccardo Paolillo, Giulio Rossi

andrea.detti@uniroma2.it
Spatial Database: Introduction

- A database system for *spatial object*
- Spatial objects are spatial structures like Point, MultiPoint, Lines, Polygons with associated properties

```json
"Type": "Point"
"Coordinates": [-77.0461, 38.9163]
"Properties":
  { "name": "Washington Hilton"}
```
Spatial Database: Introduction

- Spatial database offers additional capability to query spatial objects
- **Inclusion**: query for objects contained entirely within a specified geometry, e.g. rectangular box
- **Intersection**: query for objects that intersect with a specified geometry.
- **Proximity**: query for the objects near to a given point.

Intersect Query

```json
"Type": "Point"
"Coordinates": [-77.0461, 38.9163]
"Properties":
  { "name": "Washington Hilton"}
```

Query Answer

```json
"Type": "Point"
"Coordinates": [-77.0467, 38.9162]
"Properties":
  { "name": "The Churchill Hotel"}
```
Spatial Database: Introduction

- GeoJSON is a format for encoding a variety of geographic data structures in a JSON structure
- Widely used in mobile apps and Javascript, RFC 7946
- GeoJSON supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon.

```json
{
    "geometry": {
        "type": "Point",
        "coordinates": [-77.0461, 38.9163]
    },
    "properties": {
        "name": "Washington Hilton"
    }
}
```
Spatial Database: Introduction

- Spatial Database could have a centralized deployment or a distributed deployment
- Distributed deployment = horizontal scalability
  - Do you need performance? Just deploy another DB engine
- Front-end
  - Query routing, Access Control
- DB Engine
  - Query processing
  - Storage, Caching
SPATIAL DATABASE OVER ICN
OpenGeoBase

• Distributed/Federated Spatial Database based on **Information Centric Networking (ICN)** technology
Why? Which are ICN benefits?

**ICN DB**

- **Routing by name:**
  - Geographical sharding
    - DB engines associated to different geographical partition
    - A DB engine can be dedicated to store all data of a country
  - Query routing
    - Queries are sent only to relevant DBs intersecting the requested area

- **Data-centric security**
  - Data-level access control
    - Within a same table the data owner can Read/Write its data item and can only Read data item of others
  - Data owners are responsible for data validity (signature inside the data)

- **Federated deployment**
  - Many administrators responsible of their data partition

**Existing noSQL DB (e.g. MongoDB)**

- **Hash routing**
  - Hash sharding
    - DB engines associated to different hash partition
    - A DB engine can not be dedicated to store data of a country
  - Query flooding
    - Queries sent to all DBs

- **Table-centric security**
  - Table-level access control
    - Within a same table user has the same right on all the table data
  - Administrator is responsible for data validity

- **Distributed deployment**
  - Single administrator responsible for all
DB model

- NoSQL <Key, Value>, i.e. <Name, Content>
- Each GeoJSON spatial object is stored in the DB as a ICN Content with a unique name

ICN Content

```json
{
  "geometry": {
    "type": "Point",
    "coordinates": [-77.0461, 38.9163]
  },
  "properties": {
    "name": "Washington Hilton"
  }
}
```
Query-Response

- Query = Interest
- Response = Content Object
Object query

- Give me object ID ndn/OGB/...1234
- Straightforward
Range query

• Give me all objects intersecting this area box \([[[SW] [NW], [NE] [SE]]]\)

• Not trivial as before

• The Interest name does not represent an object but a condition

• Response is not a single object but a collection of objects

Q: How can we support range queries over ICN in an effective way?

A: Through a spatial index.
Range query and indexes

- Spatial DB use an additional internal structure: the *spatial index*
- OGB uses *three layer hierarchical-grid spatial index*
- Each *TILE* has a unique ICN *tile-prefix*, including its GPS coordinates
Index Objects

• For each GeoJSON spatial object, there exists also as many index objects as the number of intersecting tiles of the grid.

• The name of the index object include the tile-prefix, and its content is the name of the spatial object it refers to.
Tile Objects

• A tile can contain several Index Objects

• A Tile Object is a container of all these index objects; its name includes the tile-prefix; it is **built at run time** by the DB engine.
Content Objects and Naming Schemes

<table>
<thead>
<tr>
<th>Name</th>
<th>Content</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>ndn:/OGB/77/38/GPS-ID/GeoJSON/Hilton/1234</td>
<td>{&quot;geometry&quot;: {&quot;type&quot;: &quot;Point&quot;, &quot;coordinates&quot;: [-77.0461, 36.9163]}, &quot;properties&quot;: {&quot;name&quot;: &quot;Washington Hilton&quot;}}</td>
<td>Hilton Sign</td>
</tr>
</tbody>
</table>

Spatial Object

| ndn:/OGB/77/38/GPS-ID/DATA/Hilton/1234 | Hilton Spatial Object Name | Hilton Sign |
| ndn:/OGB/77/38/GPS-ID/DATA/Churchill/3210 | Churchill Spatial Object Name | Churchill Sign |

Index Object

| ndn:/OGB/77/38/GPS-ID/TILE | Admin Sign |

Tile Object
Geographical Sharding

• Each Database Engine stores only Objects related to a set of tiles it is responsible to

• Related tile-prefixes are advertised through NLSR

ndn:/OGB/77/38/GPS-ID/

Advertisement

Level-0 index tiles
100x100

Level-1 index tiles
10x10

Level-2 index tiles
1x1

World
Range query processing: tessellation

- Query-handler: Tessellation + Index Fetch + Spatial Object Fetch + Postfiltering
- Tessellation: identification of a “small” number of tiles of the index covering the requested area
Range query processing: fetch

- Index fetch: fetching of index elements using *tile queries*
  - ICN routing-by-name for query-routing
- Spatial Object fetch: fetching of GeoJSON spatial objects referenced by the index elements contained in the tile objects

Diagram:

- Mobile device
- Query handler
- ICN
  - Tile-query 1 (Interest)
  - Tile-query 2 (Interest)
  - Tile-data 1 (Data)
  - Tile-data 2 (Data)
  - GeoJSON-query 1 (Interest)
  - GeoJSON-data 1 (Data)
Data Insert

- An Insert-Handler package the Spatial Object and related Index Objects
- Discovers and caches the IP addresses of the DB engines responsible of the related tiles through an ICN-based address resolution
- Push Content Objects through a TCP/IP socket
- Temporary solution, but very fast (one-way delay)
Multi-Tenancy and Security

- Admin grant access to tenants.
- Tenant grant access to users.
- User can
  - Read only data inserted by other users of the same tenant
  - Create/Read/Update/Delete own data
- Each user has a Private key and a Digital Certificate with public key, signed by its Tenant.
- Same for Tenant, but Admin signs
- Admin is the final thrust anchor
- Policy enforcement through “simple” configuration of NDN Validator
Use case

• EU Intelligent Transport System forthcoming Directive 2010/40/EU requires the setting up of a single National Access Point (NAP) and its associate “discovery/search and browse” functionality for national ITS services, by each Member State

• Each Nation could have an own OGB site for discovery services by spatial queries.

• OGB stores spatial objects referencing ITS data
  – e.g. a multipoint object can be used to link to GTFS data sources
Performance evaluation

(a) Batch tile-query duration vs. number of tile-queries for 1x1 and 10x10 tiles, in case of 1 and 4 cache hit probability, for 10x10 tiles, in case of 1 and 4 database engines, no BF

(b) Duration of a batch of 500 tile-queries time vs. cache hit probability for 1x1 and 4 DBs

(c) Range-query time vs. range query area, 4 database engines, no BF, constrained tessellation with k max-tiles, with and without ICN cache of 5000 items

(d) Red dots indicate not-void tiles for the GTFS application

(e) Average duration of range-query, query-tiles

(f) Average number of tiles vs. range-query area for batch, tessellation and BF request vs. range-query the GTFS application area for the GTFS application
Conclusions

• ICN functionality such as routing-by-name and data-centric security make possible to realize spatial databases with features that are difficult to obtain using off-the-shelf DB
  – Geo sharding
  – Query routing
  – Data-level security
  – Federation

• Be careful with caching, stale data are not acceptable in DB environment
  – OGB deployment only cache in the DB engine and we modified NDN Repo to internally support caching with invalidation

• OGB is evolving
  – We are changing a bit the indexing rules to save space
  – We are designing new insert scheme exploiting Link Objects
  – We are going to extend GeoJSON support to polygons, etc.
Thank you
Questions?

UNIVERSITY OF ROME “TOR VERGATA”
Department of Electronics Engineering
Via del Politecnico, 1 - 00133 Rome - Italy

Andrea Detti, Ph. D.
Professor of Telecommunications

Phone: +39 06 7259 7445
e-mail: andrea.detti@uniroma2.it
Fax: +39 06 7259 7435
http://netgroup.uniroma2.it/Andrea_Detti
Backup slides
import com.bonvoyage.ogb.client.*;

String uid = "admin"; // user id
String tid = "bonvoyage"; // tenant id
String pwd = "ogb"; // password
String cid = "GTFS"; // collection id
String serverURL = "https://160.80.103.207:443";
String token;

OgbClient ogbTestClient = new OgbClient(serverURL);

// LOGIN
token=ogbTestClient.login(uid, tid, pwd);
Programming: Insert a GeoJSON Point

// INSERTION OF POINT OBJECT
// point coordinates (lon,lat)
double [] coordinates = {0.1, 0.1};

// point properties
HashMap<String,String> prop = new HashMap<String,String>();
prop.put("train-name", "ICE 373");
prop.put("train-speed", "170 km/h");

// db insertion, response is the object identifier (oid)
String oid = ogbTestClient.addPoint(token,cid, prop, coordinates);
// RANGE QUERY, response is a JSON Array of GeoJSON objects
double sw_lat=0.0;
double sw_lon=0.0;
double size = 0.5;
String response = ogbTestClient.rangeQuery(token, cid, sw_lat, sw_lon, size);
Performance

$K = \text{max number of tiles used for tessellating the range query area}$

![Graph showing performance with different values of $k$ and with and without cache.]
Conclusions

• High User Experience:
  – in-production horizontal scalability, Load balancing, Caching

• High Availability:
  – Replication with automatic failover

• Secure:
  – Access control with user permissions, Cyphering (HTTPS), Data-Centric Security

• Multi-tenants multi-users
  – Tenant=application owner
  – User=data owner, that can access data of same tenant users

• Versatile Application Frameworks:
  – GeoJSON
  – HTTP interface

• Simple DB federation

• Simple Programming Interface
Technology

- **Data bases connected by an Information Centric Network (ICN)**
- ICN nodes route DBs query towards the right DB using its **forwarding by name** functionality
  - A query is an Interest packet
  - An answer is a Data packet
- ICN nodes **cache** answers on memory for accelerating lookup
- ICN nodes carry out **access control** on query
Activity overview

• OpenGeoBase
  – NoSQL distributed data base optimized for geo referenced data
  – ICN networking

• Travel Centric Services
  – HTTP/ICN services for sharing travel data among transport info providers and travel services providers (travel operator, etc.)
Travel Centric Services

• Specific Services for BONVOYAGE activities
• Exploit OpenGeoBase storage
• Expose simplified API (HTTP/NDN) to the users
  – transport information provider
  – travel service provider
Travel Centric Services: Data Insert

• Transport Information Providers can insert **geolocated references** to their data and services

• E.g.: A Transport Provider dispose of a GTFS file with a stop in 42.12345N 13.45678E
  – Store the URI in the related OpenGeoBase tiles of layer 0,1,2
  – \(<\text{tiles id, tenant}_\text{id, user}_\text{id, key, value}>\)
    • l0: \<(42,13), bonvoyage, trenitalia, GTFS/TRAIN, www.trenitalia/....>\)
    • l1: \<(42.1,13.4), bonvoyage, trenitalia, GTFS/TRAIN, www.trenitalia/....>\)
    • l2: \<(42.12,13.44), bonvoyage, trenitalia, GTFS/TRAIN, www.trenitalia/....>\)
Travel Centric Services: Info Discovery

• Travel Service Operators can select an *Discovery Area* (GPS box) and Get the values of the keys associated to the area

• E.g. Selecting an area containing any stops of Trenitalia GTFS file, the Travel Service Operator obtains the URI of the Trenitalia GTFS file, as well as of any other GTFS file with stops in the area

• GTFS is an example, any other Transport information URI can indexed and searched in this way
Travel Centric Services: Info Discovery

Travel Centric Services
Travel Centric Service: Architecture

HTTP
- Travel Service Provider
- Web Server
- Application Server (JAVA EE Spring MVC)

ICN (NDN)
- OpenGeoBase
HTTP API

• Server URL:
  – http://cloud.netgroup.uniroma2.it/Bonvoyage/mapservlet

• HTTP POST with JSON content
  – coordinates: (mandatory) the GPS coordinates of the NE and SW point of the requested area;
  – tenantId=bonvoyage: (mandatory) id of the tenant;
  – dataType: (mandatory) specifies information type requested (now only “GTFS”);
  – options: (optional) a list of values of parameters for filtering the request depending on dataType.

  • For GTFS dataType it is a list of values [x,y,z,...], where each value represent the GTFS route type user is interested in: 0:Tram, Street Car and Light Rail; 1: Subway and metro; 2: Rail; 3: Bus; 4: Ferry; 5: Cable car; 6: Gondola, suspended cable car; 7: Funicular; void list means all
HTTP API

- **resolution**: (optional) specifies the resolution of the area-tiles used to represent the requested area. Values are 0: 100 km x100 km, 1: 10 km x10 km, 2: 1 km x 1 km. A greater resolution (e.g. 100x100 vs 1x1) reduces query time but the actual discovery area may be much greater than then requested area.

- **maxTiles**: (optional) specifies the maximum number of area-tiles used to represent the requested area. It is alternative to resolution parameter (not used in presence of resolution) and the server automatically computes the resolution of the area-tiles. Default value is 50.

- **command** = **LIST**: (mandatory) action to be performed by backend server

- **format** = 1: for future use
HTTP API

• HTTP Response JSON Object with these fields:
  – *tiles*: the set of OpenGeoBase tiles upon which discovery has been actually carried out
    • northEst: coordinate of the north est corner of tile.
    • southWest: coordinate of the south west corner of tile.

• uriList: the list of URIs of the Transport Information Provider resources discovered in the set of tiles.
  – *ndnName*: the ndn name of the file.
  – *httpUrl*: the http URL of the file.
Interaction with OpenGeoBase

• Mapping Problem:
  – user requests data discovery for a “random” discovery area
  – OpenGeoBase (OGB) makes possible to query fixed tiles and not random areas

• Solution
  – The App server covers the discovery area with a set of OGB tiles
  – Then, query tiles to OGB, gets values from OGB and send back via HTTP
Interaction with OpenGeoBase

• Mapping Optimization Problem: the more the tails of the covering sets, the higher the number of queries, with all related drawbacks
  – Choosing covering tiles of 1x1 km (layer 2) may provide an high number of tiles but the set of tiles cover an area very close with the discovery area selected by the use
  – Choosing covering tiles with greater dimension (e.g. layer 0,1) decrease number of tiles of the set, but the actual discover area may be rather greater than the requested one

• Solution: in progress, now API make possible to select the max number of tiles and use the smaller possible single resolution
On going

- Now only few GTFS files. We are going to upload **800** GTFS files crawled by google :-)
- Alpha release of Production Software for Transport Service Provider for Publishing GTFS Data by own coming soon
- OpenGeoBase do not support access control now ... we are finalizing security framework and API
- Mapping optimization with variable resolution
- Real time support
- Interaction with ITS cluster
- More....
Andrea’s Bird Eyes View

Bonvoyage App

Orchestrator #1
(Bonvoyage)

Orchestrator #2

Travel Service Providers

Users

OpenGeoBase/TravelCentricServices
Indexing

Transport Information Provider

National Routing Engine

National Routing Engine

National Routing Engine

National Point of Access (NPA) (Data and Services)

Relevant Standardization Impact

NPA standard interface
ICN re-thinks the role of the OSI network-layer

No more send data to hosts identified by an address, but provide hosts with information identified by names

Network packet header includes the name of the requested/transported information so making aware network routers of “what” they are handling

This awareness makes easy to implement in-network: content based functionality
  – Routing-by-name (handle replications)
  – In-network caching
  – Multicast

These functions strongly simplifies development of apps on top of the ICN API
Information Centric Networking: CCN Data Model

• Several architectures ... V. Jacobson Named Data Network (NDN) is likely the topmost referred, implemented by NDN (named-data.net)

```
hierarchical names
/foo.eu/video1/SN=1/BW=100.mp4/$cn1
```

```
Content
/foo.eu/video1/SN=1/BW=100.mp4
```

```
Data message
Chunk Name
Data Chunk
```

```
Interest message
Chunk Name
```
**Information Centric Networking: Node Model**

### Cache (content store)

<table>
<thead>
<tr>
<th>Name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>/foo.eu/video1/SN=1/BW=100.mp4/$cn1</td>
<td>...</td>
</tr>
</tbody>
</table>

### Forwarding Information Base (FIB) (prefix match)

<table>
<thead>
<tr>
<th>Name</th>
<th>Face</th>
</tr>
</thead>
<tbody>
<tr>
<td>/foo.eu/video1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Pending Interest Table (PIT)

<table>
<thead>
<tr>
<th>Name</th>
<th>Requesting Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>/foo.eu/video1/SN=1/BW=100.mp4/$cn1</td>
<td>0 1</td>
</tr>
</tbody>
</table>
File Transfer: receiver driven TCP

Receiver

<table>
<thead>
<tr>
<th>cwnd=1</th>
<th>Interest “cnn.com/text1.txt/chunk1”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data“cnn.com/text1.txt/chunk1”</td>
</tr>
<tr>
<td></td>
<td>Interest “cnn.com/text1.txt/chunk2”</td>
</tr>
<tr>
<td></td>
<td>Interest “cnn.com/text1.txt/chunk3”</td>
</tr>
<tr>
<td>cwnd=2</td>
<td>Data“cnn.com/text1.txt/chunk2”</td>
</tr>
<tr>
<td></td>
<td>Data“cnn.com/text1.txt/chunk3”</td>
</tr>
<tr>
<td>cwnd=3</td>
<td></td>
</tr>
</tbody>
</table>
Real Time: give me next

Receiver

Interest “alice/voip/ts=1”

Data“alice/voip/ts=1”

Interest “alice/voip/ts=2”

Data“alice/voip/ts=2”

Speaker
NDN

• named-data.net
• open source implementation of NDN model
• It is very well supported by NDN project
• Java library for application development (Linux and Android)
• Node model implemented in C (Linux and Android)
Information Centric Networking : Contents

- Each content (file, query response, etc.) has a *hierarchical* name
- Is chunked in Data Packets, whose name includes the chunk number

```
/OGB/Tile1/index/%01
```

```
Content

Data packet

Chunk Name

Data Chunk

Interest packet

Chunk Name
```
Horizontal Scalability – Data Replication

• Different engines manage the same geographical tiles
• ICN routing-by-name used to balance queries and insertion
• Sync proto used to synchronize DB engines
• NLSR routing protocol used for routing configuration
Data and naming schemes

• We have three main ICN Content Objects in the DB
• **OGB-GeoJSON**, storing the actual GeoJSON spatial data
• **OGB-Data**, storing the index data related to a GeoJSON spatial data
• **OGB-TILE**, built at run-time by the engine and embedding the list of Index Data that are related to a TILE
  – more object could be in a same tile, more index data have be sent back
• Each name start with a tile-prefix that uniquely identifying the referenced tile through GPS coordinates. Thus we can carry out Geographical Sharing through routing by name

  ndn:/OGB/lng(0)/lat(0)/lng(1)lat(1)/.../lng(n)lat(n)/GPS-ID
Range query and index

- To speedup range query, spatial databases use an additional internal structures, named *spatial index*
  - For each data item the database stores the item and some metadata referencing the data item within index tables

We need to insert in the ICN DB also Index Elements, not only spatial object
Horizontal Scalability – Geo Sharding

- Different engines manage different geographical zones/tiles
- ICN routing-by-name used to steer queries and insertions towards proper engine
- NLSR routing protocol used for routing configuration