END-TO-END GRAPH MAPPER

Benjamin Billet, Mickaël Jurret, Didier Parigot and Patrick Valduriez

EPI **ZENITH**

Inria Sophia Antipolis Méditerranée

33ème conférence sur la Gestion de Données
Principes, Technologies et Applications
BDA 2017

- Startup which develop/market **mobile social networks for micro-communities**
  - Cross-platform (Android, iOS, web), specific-purpose (event, activities)

Triton Goals

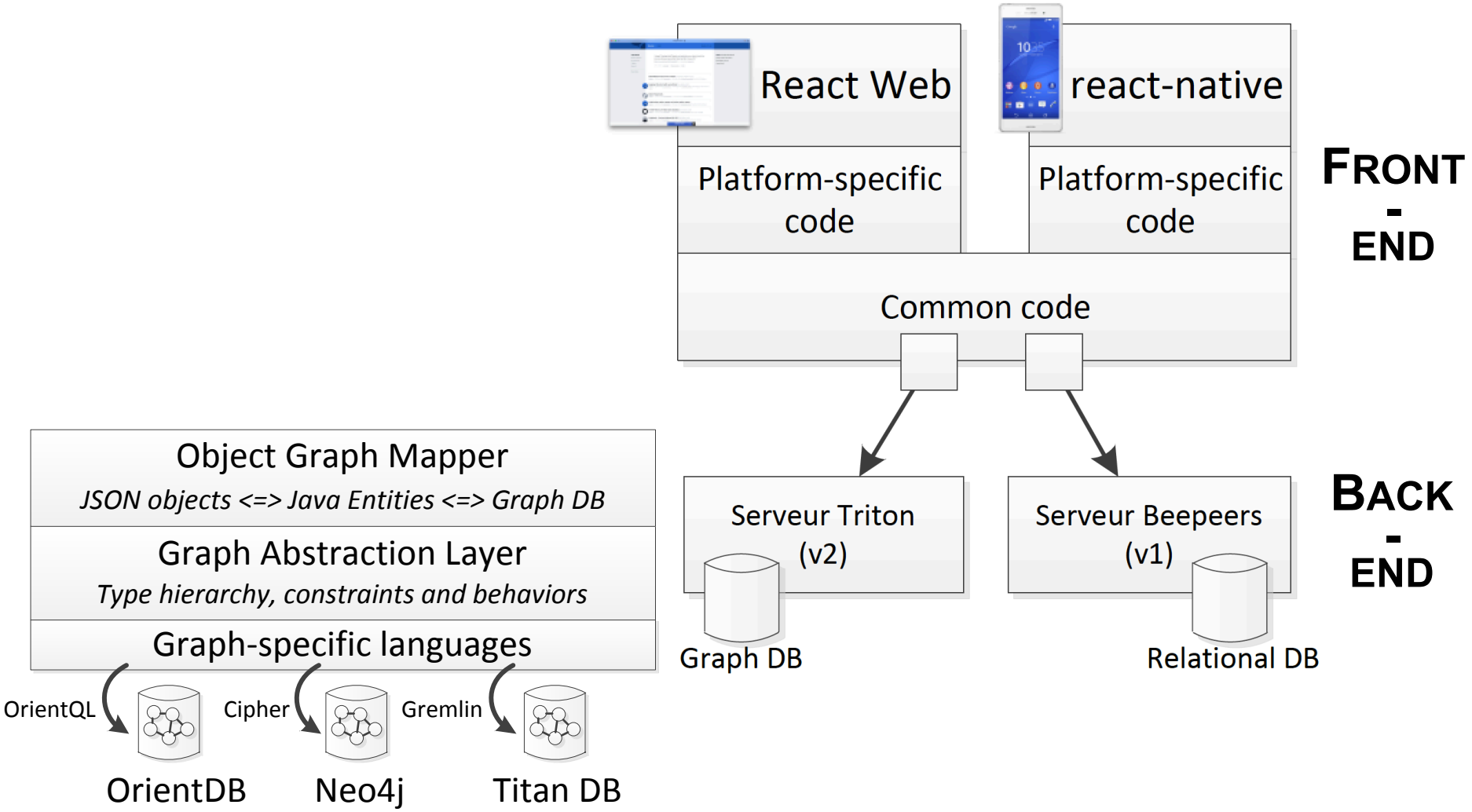
- Current state: A software factory for building customized social networking servers and databases, according to customers' requirements
  - Configuration/specialization over development
  - Rapid and adaptable production
  - Scalability

Triton Goals

- Current state: A software factory for building customized social networking servers and databases, according to customers' requirements
  - Configuration/specialization over development
  - Rapid and adaptable production
  - Scalability

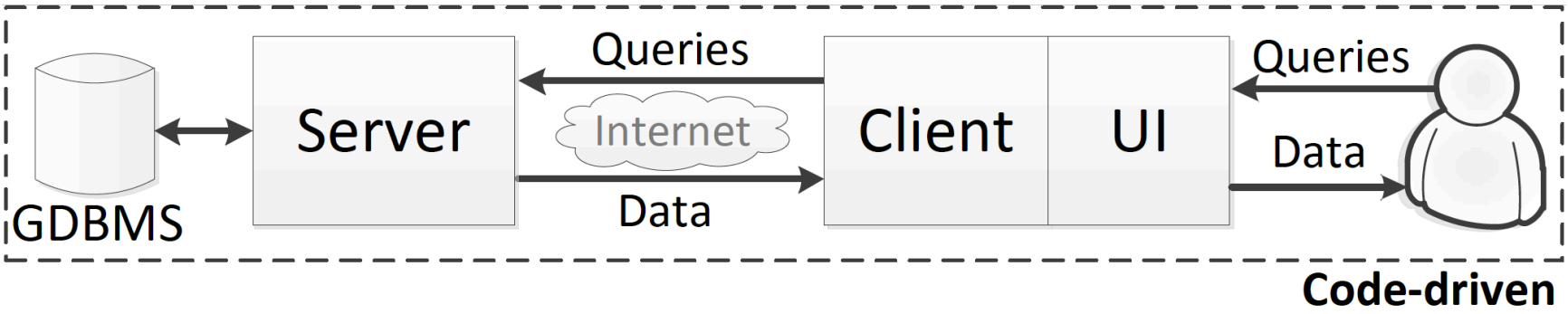
- Next step: Building mobile clients as well and generalize the approach to various applications that manage linked data

Current Infrastructure



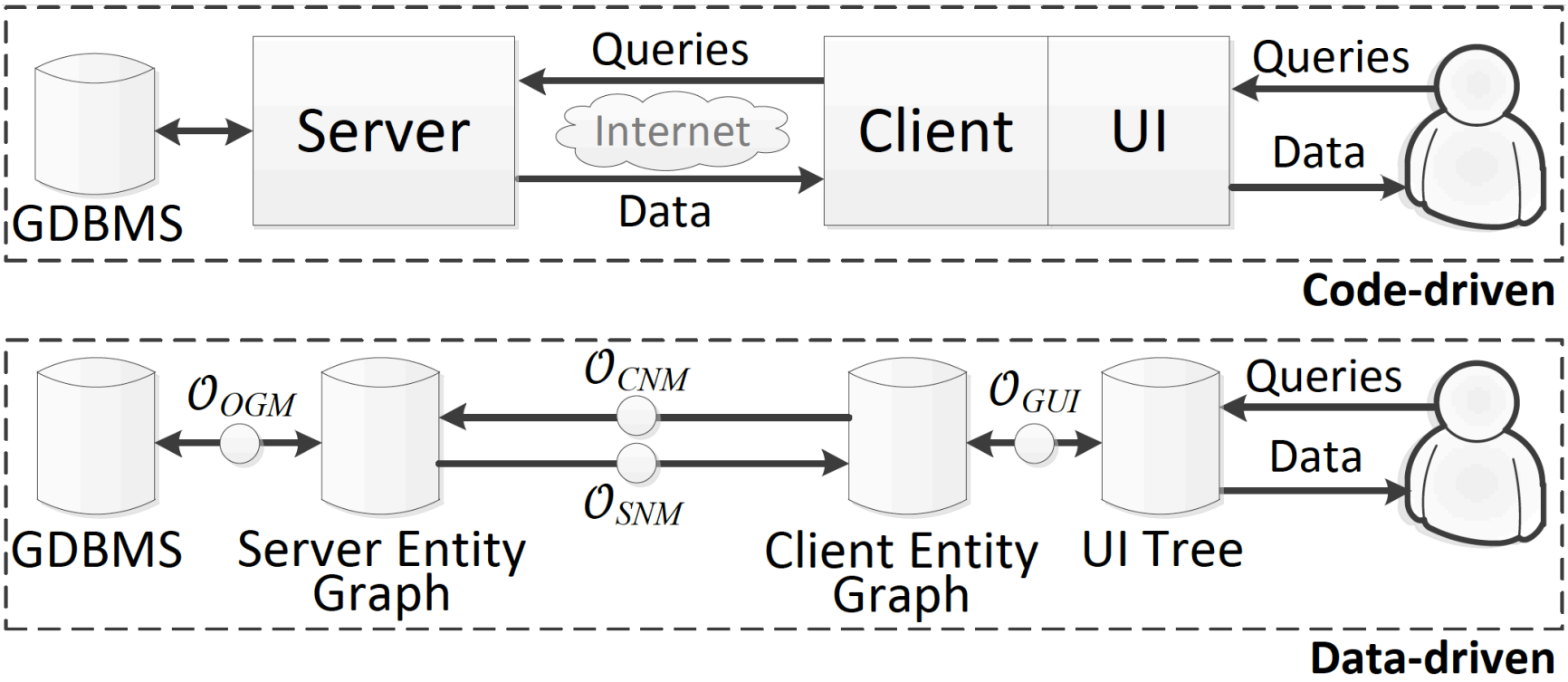
Typical Web and Mobile applications

- Client-server
- CRUD on a set of connected data



Typical Web and Mobile applications

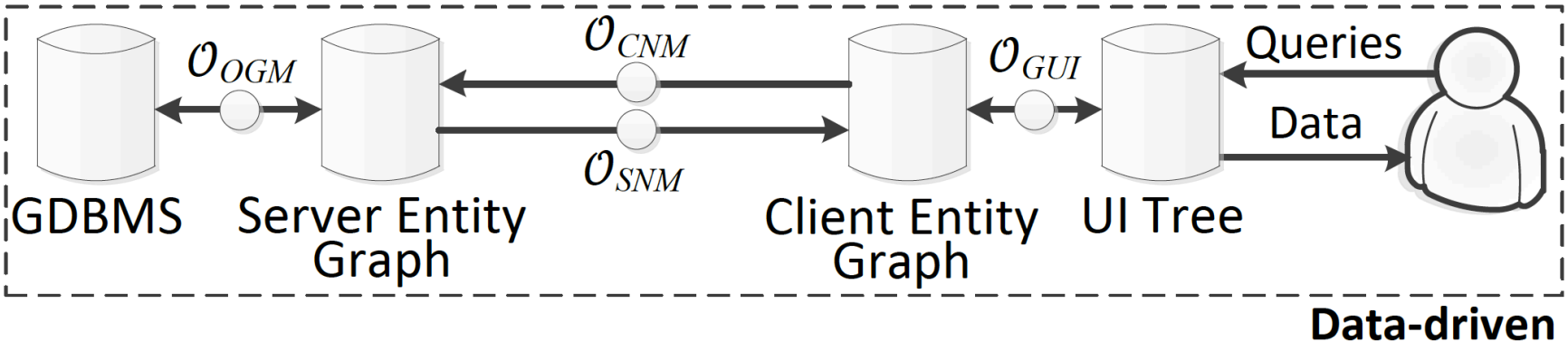
- Client-server
- CRUD on a set of connected data



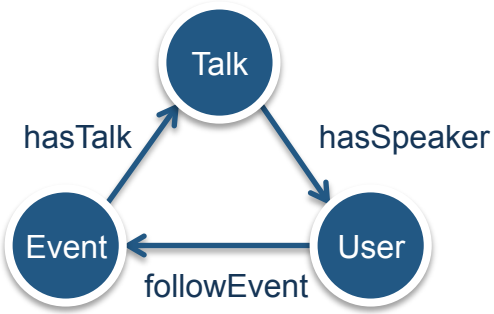
Typical Web and Mobile applications

- Client-server
- CRUD on a set of connected data

$\mathcal{O}_{GUI}$ UI Tree | Client Graph Mapping Operator
$\mathcal{O}_{CNM}$ Client Entity Graph Queries | Web Services Call Mapping Operator
$\mathcal{O}_{SNM}$ Server Entity Graph | Web Services Response Mapping Operator
$\mathcal{O}_{OGM}$ Server Entity Graph | GDBMS Mapping Operator



Graphs everywhere



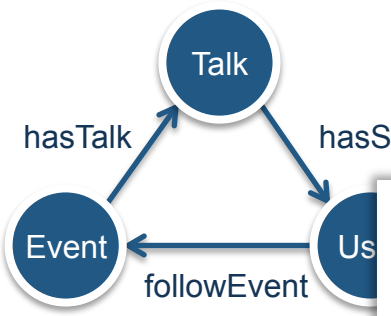
Graph Database

Event
attrs: {name, string}, {begin, date}, {end, date},
{location, string}
links: {hasTalk, Talk, one-to-many}

Talk
attrs: {name, string}, {begin, date}, {end, date}
links: {hasSpeaker, User, one-to-many}

User
attrs: {name, string}
links: {followEvent, Event}

Graphs everywhere

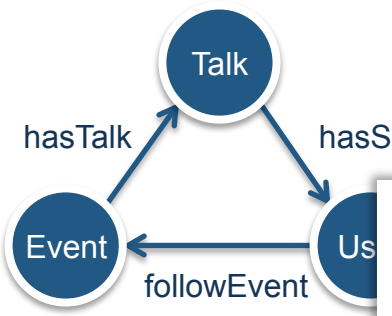


Graph Database

```
Event event = new Event();
event.setId("zenithDay");
event.setName("Zenith Day");
event.setLocation("Saint Gely");
event.setBeginDate(new Date(2016, 9, 13));
event.setEndDate(new Date(2016, 9, 13));

User user = ...;
user.getFollowedEvents().add(user.getId());

Talk talk = new Talk();
talk.getSpeakers().add(user.getId());
talk.setBegin(new Date(0, 0, 0, 11, 0));
talk.setEnd(new Date(0, 0, 0, 11, 0));

event.getTalks().add(talk);
...
```

Server Code (Java)

Graphs everywhere



Graph Database

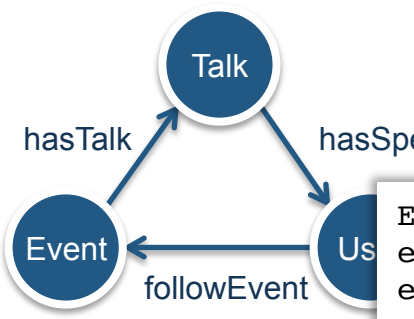
```
Event event = new Event();
event.setId("zenithDay");
event.setName("Zenith Day");
event.setLocation("Saint Gely");
event.setBeginDate(new Date(2016, 9
event.setEndDate(new Date(2016, 9,

User user = ...;
user.getFollowedEvents().add(user.g

Talk talk = new Talk();
talk.getSpeakers().add(user.getId()
talk.setBegin(new Date(0, 0, 0, 11,
talk.setEnd(new Date(0, 0, 0, 11, 0

event.getTalks().add(talk);
...
```

Server Code

JSON Serialization

```
{
  "users": [{
    "id": "bbillet",
    "firstname": "Benjamin",
    "lastname": "Billet",
    "followEvents": [ "zenithDay" ]
  }],
  "event": [{
    "id": "zenithDay",
    "name": "Zenith Day",
    "location": "Saint Gely",
    "beginDate": 1473717600,
    "endDate": 1473717600,
    "talks": [{
      "speakers": [ "bbillet" ],
      "begin": 36000,
      "end": 37800
    }]
  }]
}
```

Graphs everywhere



Graph Database

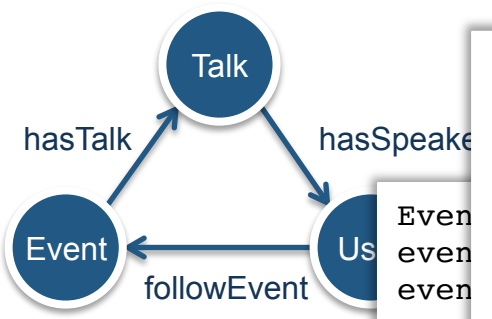
Client State

```
let store = {
  users: {
    bbillet: {
      id: 'bbillet',
      firstname: 'Benjamin',
      lastname: 'Billet',
      followEvents: [ 'zenithDay' ],
    },
  },
  event: {
    zenithDay: {
      id: 'zenithDay',
      name: 'Zenith Day',
      location: 'Saint Gely',
      beginDate: new Date(...),
      endDate: new Date(...),
      talks: [{
        speakers: [ 'bbillet', ... ],
        begin: new Date(...),
        end: new Date(...),
      }, ... ],
    }
  }
}
```

JSON Serialization

```
[{
"bbillet",
name": "Benjamin",
ame": "Billet",
wEvents": [ "zenithDay" ]

[{
"zenithDay",
: "Zenith Day",
ion": "Saint Gely",
Date": 1473717600,
te": 1473717600,
": [{
akers": [ "bbillet" ],
in": 36000,
": 37800
}]
}
```

Graphs everywhere



Client State

```
let store = {
  users: {
    bbillet: {
      id: 'bbillet',
      firstname: 'Benjamin',
      lastname: 'Billet',
      followEvents: [ 'zenithDay' ],
    },
  }
}
```

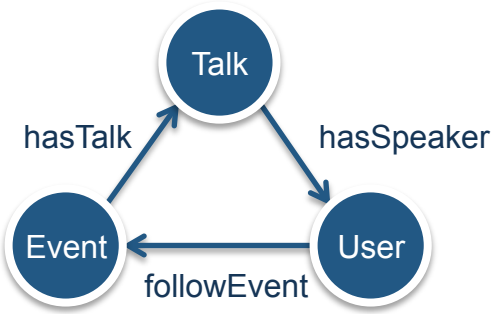
JSON Serialization

```
[{
"bbillet",
name": "Benjamin",
ame": "Billet",
wEvents": [ "zenithDay" ]

[{
"zenithDay",
: "Zenith Day",
ion": "Saint Gely",
Date": 1473717600,
te": 1473717600,
": [{
akers": [ "bbillet" ],
in": 36000,
": 37800
```

```
<View style={styles.container}>
  <View style={styles.row}>
    <Text>User name: </Text>
    <Text>
      {this.props.user.firstname}
      {this.props.user.lastname}
    </Text>
  </View>
  ...
</View>
```

Client UI (JSX)

Zoom in $\mathcal{O}_{\text{OGM}}$



Graph Database

Event
attrs: {name, string}, {begin, date}, {end, date},
{location, string}
links: {hasTalk, Talk, one-to-many}

Talk
attrs: {name, string}, {begin, date}, {end, date}
links: {hasSpeaker, User, one-to-many}

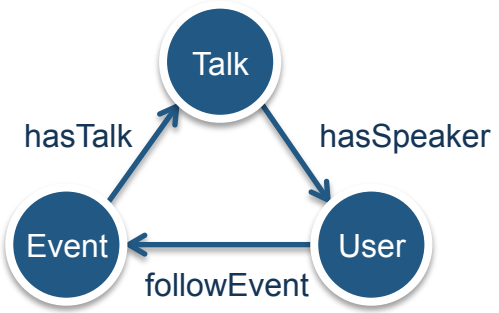
User
attrs: {name, string}
links: {followEvent, Event}

Zoom in $\mathcal{O}_{OGM}$: Mappings

Event.class
gType: Event
attrs: {title, name}, {beginDate, begin},
{endDate, end}, …
links: {talks, out(hasTalk)}

Talk.class
gType: Talk
attrs: {title, name}, {beginDate, begin},
{endDate, end}
links: {speakers, out(hasSpeaker)}, ***{event, in(Event)}***

User.class
gType: User
attrs: {name, name}
links: {followedEvents, out(followEvent)},
{followedTalks, out(hasTalk, out(followEvent))}



Graph Database

derived attributes

Zoom in $\mathcal{O}_{OGM}$: Mappings

Event.class
gType: Event
attrs: {title, name}, {beginDate, begin},
{endDate, end}, …
links: {talks, out(hasTalk)}

Talk.class
gType: Talk
attrs: {title, name}, {beginDate, begin},
{endDate, end}
links: {speakers, out(hasSpeaker)}, ***{event, in(Event)}***

User.class
gType: User
attrs: {name, name}
links: {followedEvents, out(followEvent)},
{followedTalks, out(hasTalk, out(followEvent))}

```
class User
{
    private String name;
    private Set<Event> followedEvents;
    private Set<Talk> followedTalks;
}
```

Server Code (Java)

Current State & Problems

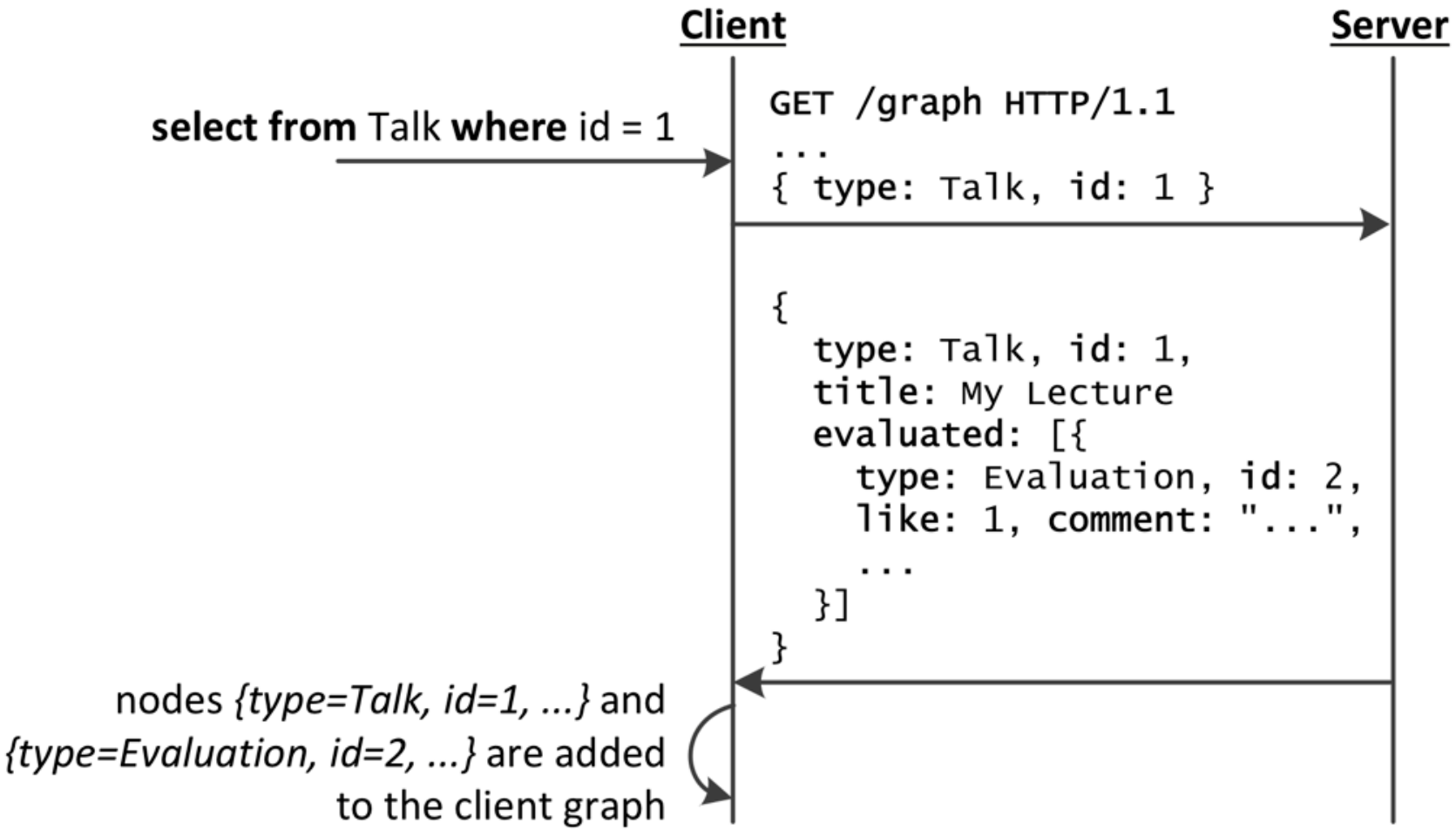
- Current state: Prototypal implementation based on OrientDB, GraphQL, redux, react-native
  - A lot of custom code for the client-side
  - Technical problems: graph databases heterogeneity and instability

Current State & Problems

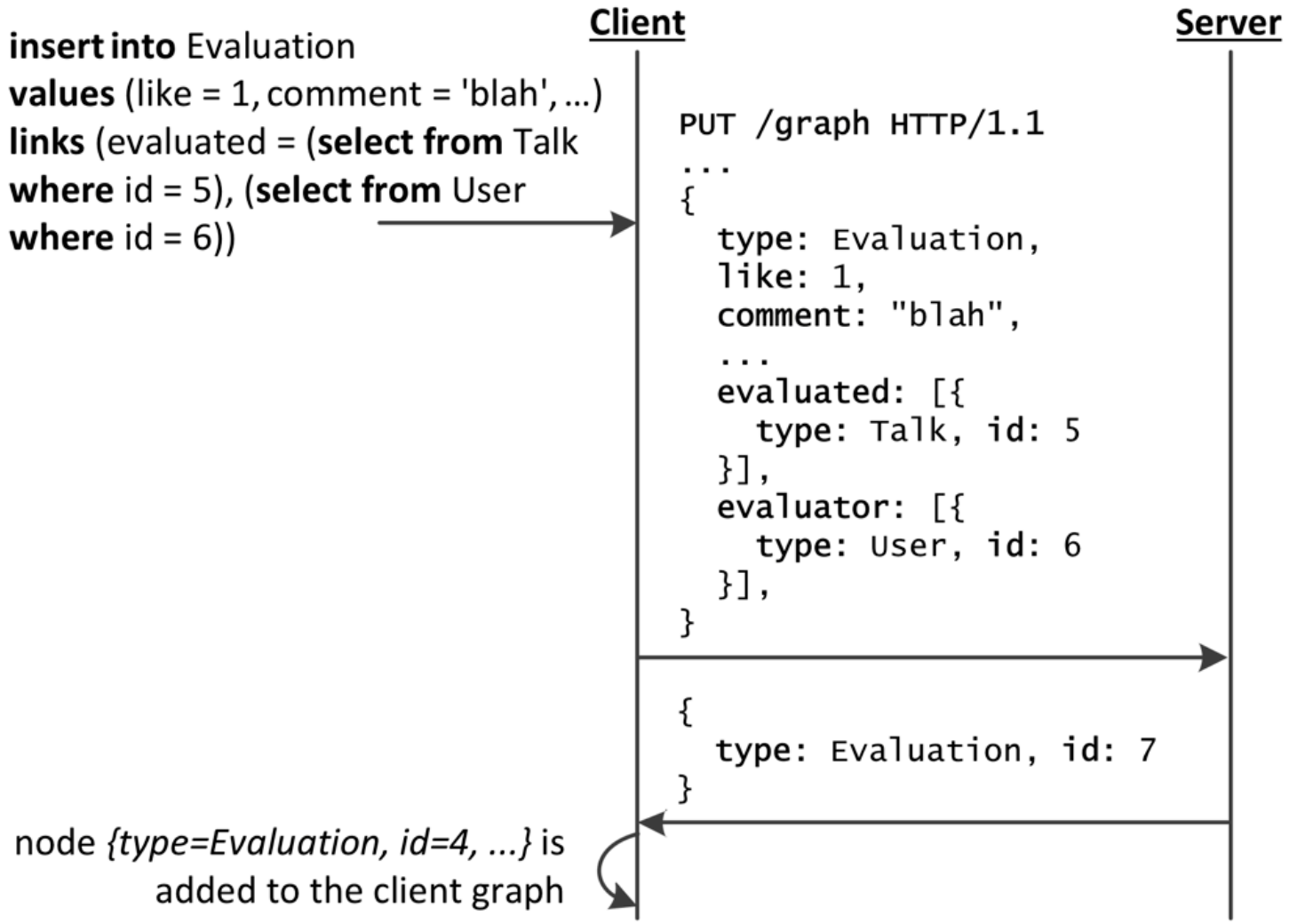
- Current state: Prototypal implementation based on OrientDB, GraphQL, redux, react-native
  - A lot of custom code for the client-side
  - Technical problems: graph databases heterogeneity and instability
- Ongoing research problems:
  - Complex mappings, identifying common mapping patterns in actual applications
  - Horizontal concerns: access control, transactions
  - Common query language for each layer (SPARQL?)
  - Distributed systems? Non-graph databases?

www.inria.fr

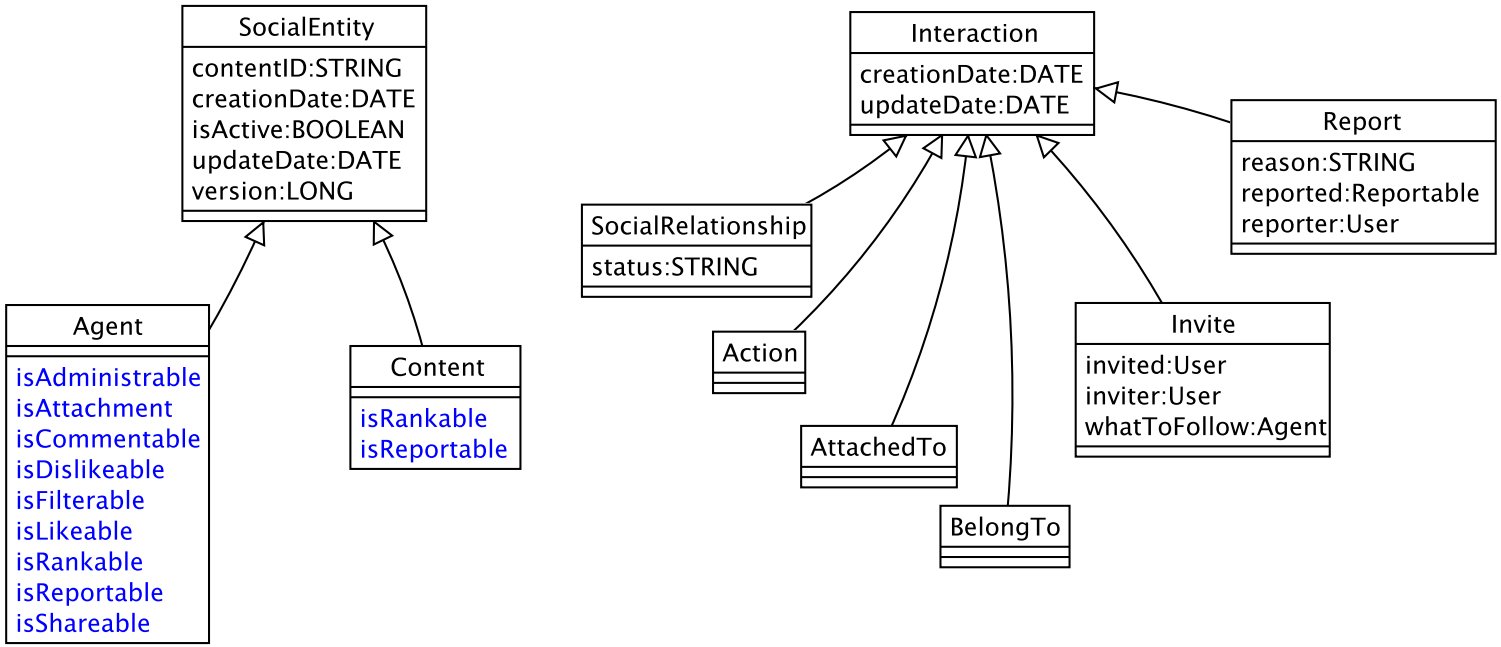
Zoom in $\mathcal{O}_{CNM}$ and $\mathcal{O}_{SNM}$: Network



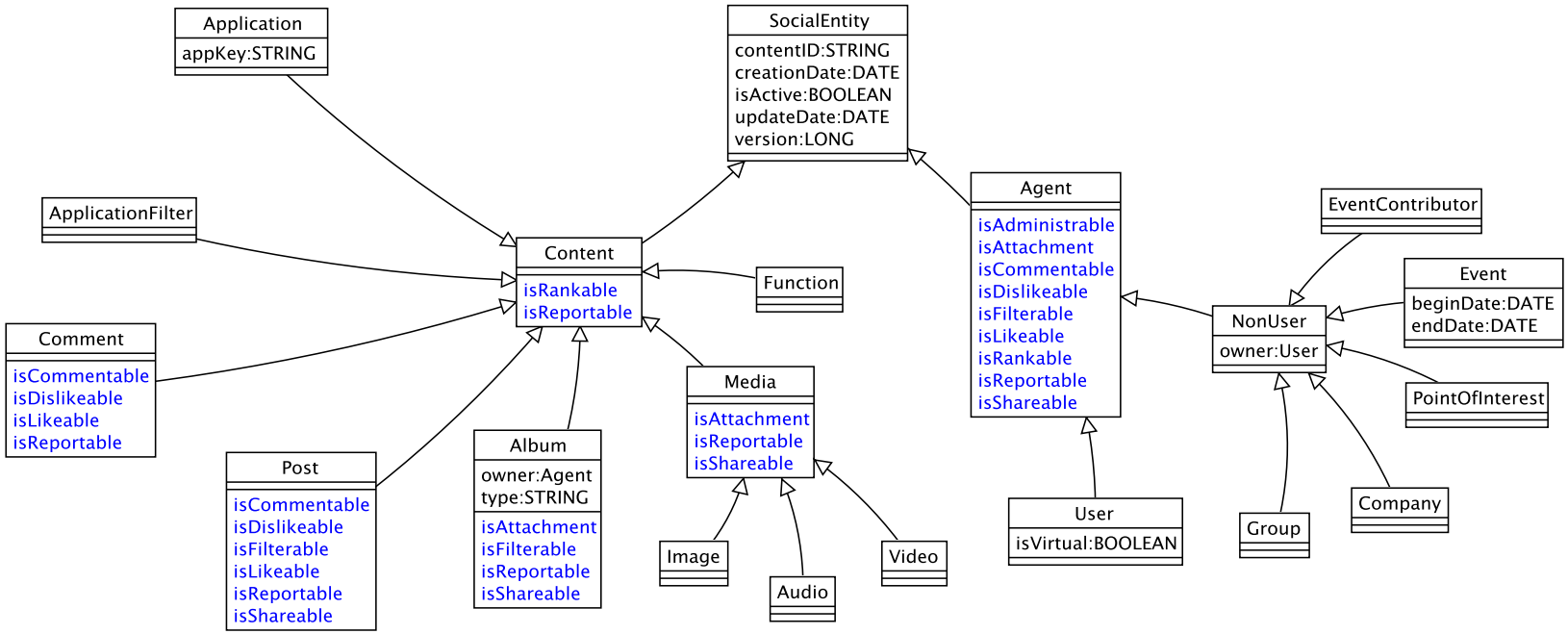
Zoom in $\mathcal{O}_{CNM}$ and $\mathcal{O}_{SNM}$: Network



Full Graph Schema: Root Types



Full Graph Schema: Social Entity Types



Full Graph Schema: Interaction Types