Tutorial on RDF Stream Processing 2016

M.I. Ali, J-P Calbimonte, D. Dell'Aglio,
E. Della Valle, and A. Mauri

http://streamreasoning.org/events/rsp2016

Insight

Hes·so VALAIS WALLIS

Universität Zürich[UZH]

POLITECNICO MILANO 1863

How to publish RDF Stream with TripleWave

Andrea Mauri

andrea.mauri@polimi.it

@janez87

What is TripleWave?

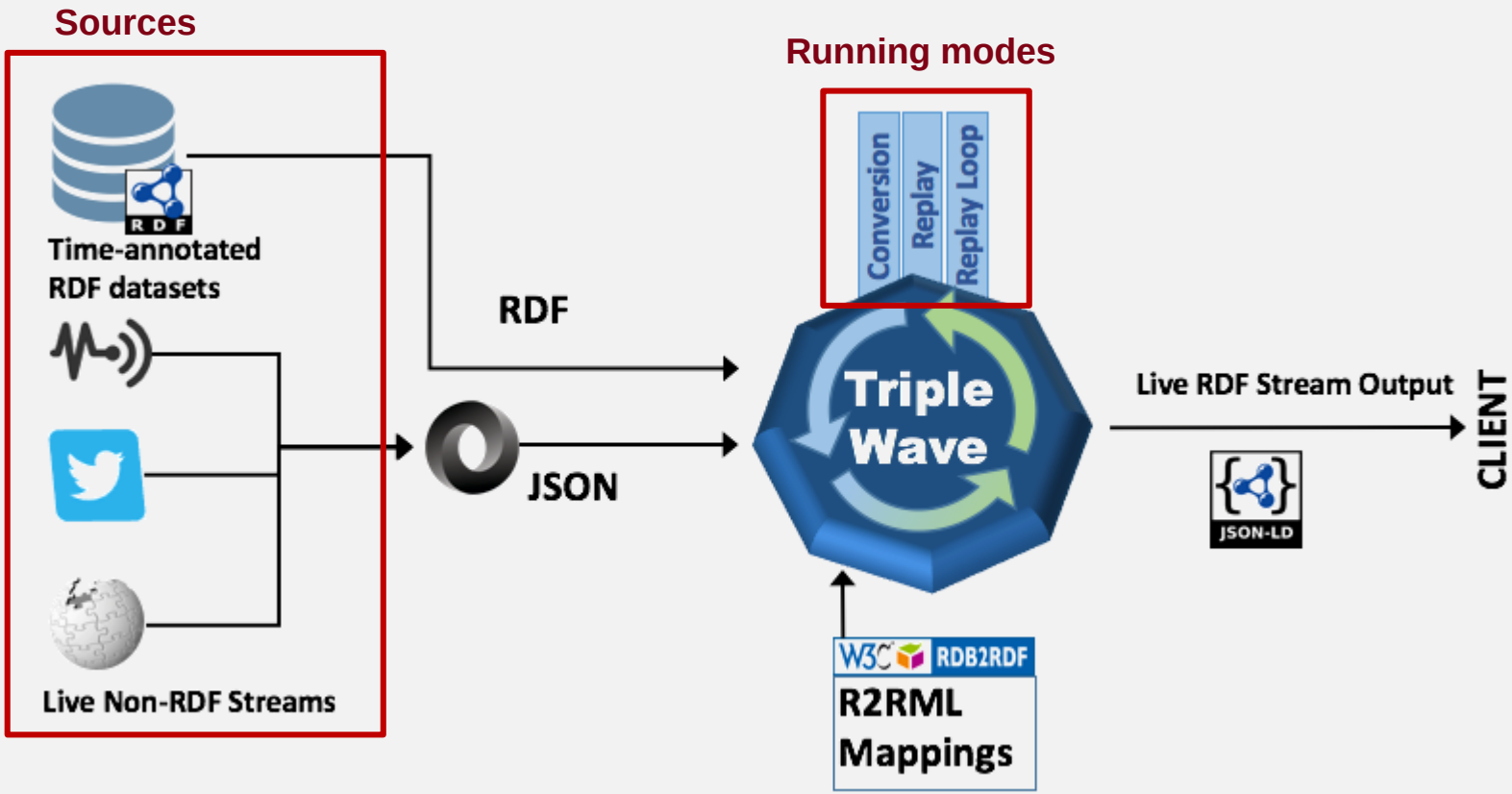
TripleWave an open-source framework for creating RDF streams and publishing them over the Web.

Why?

- Even though processing data streams is increasingly gaining momentum standard protocols and mechanisms for RDF stream exchange are currently missing.
  - Limiting the adoption and spread of RSP technologies on the Web.

- There is still a need for a generic and flexible solution for making RDF streams available on the Web

High Level Architecture

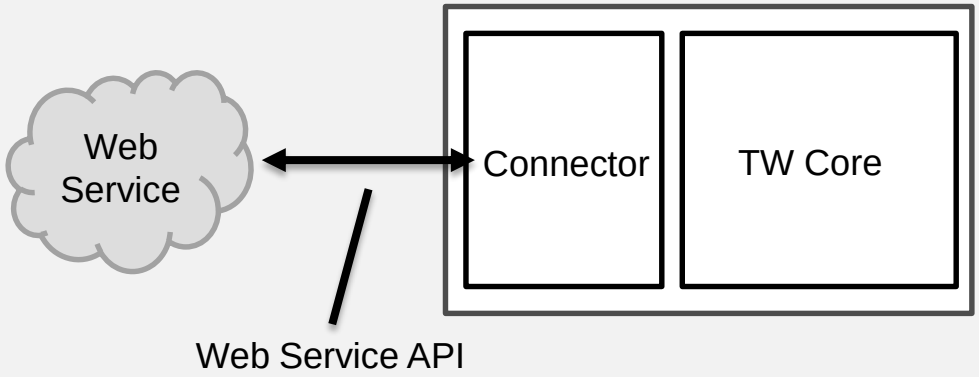


Time-annotated RDF Datasets / Replay and Replay Loop

- RDF data is available as **Linked Data endpoints** or as **simple files**.
- Convert static dataset into a continuous flow of RDF data, which can then be used by an RDF Stream Processing engine.
  - The data is published according the original timestamp (i.e. the time between triples is preserved)
- **Use Cases:** include evaluation, testing, and benchmarking applications, as well as simulation systems.

Live non-RDF Streams / Conversion

- Existing streams can be consumed through connectors



- TripleWave constructs RDF triples that will be output as part of an RDF stream
  - It uses R2RML to define the mapping that builds the RDF triples
- **Use Case**: publishing new RDF Stream

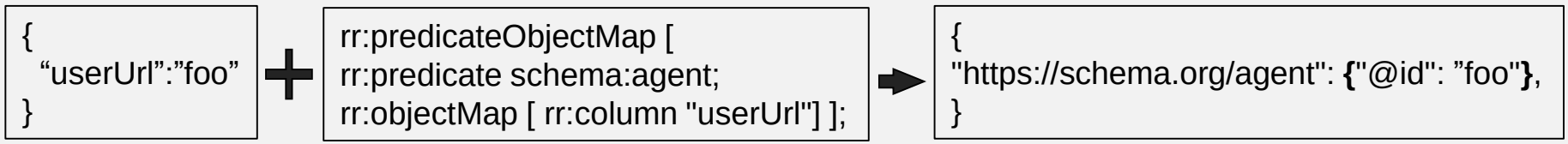
Live non-RDF Streams / Conversion - R2RML

R2RML is a language for expressing customized mappings from relational databases to RDF datasets.

We use it to map the general structure of the data to a RDF Triples

In particular you can:

- Map data field to triple field



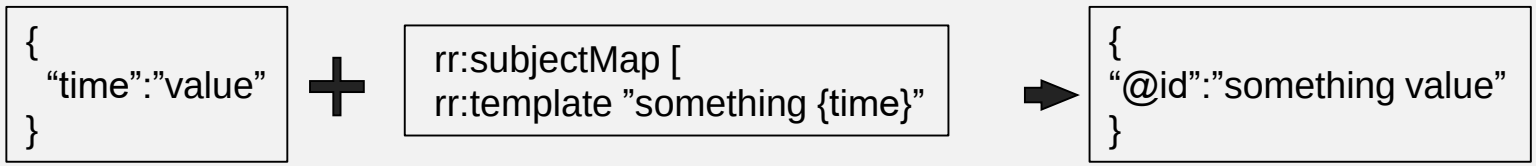
Live non-RDF Streams / Conversion - R2RML

R2RML is a language for expressing customized mappings from relational databases to RDF datasets.

We use it to map the general structure of the data to a RDF Triples

In particular you can:

- Map data field to triple field
- Map data field to triple field using a template



Live non-RDF Streams / Conversion - R2RML

R2RML is a language for expressing customized mappings from relational databases to RDF datasets.

We use it to map the general structure of the data to a RDF Triples

In particular you can:

- Map data field to triple field
- Map data field to using a template
- Add a new consant field

```
  rr:predicateObjectMap
[    rr:predicate rdf:type;  rr:objectMap
[ rr:constant schema:UpdateAction]];
```

```
{
"http://www.w3.org/1999/02/22-rdf-syntax-ns#type":
                    {"@id": "https://schema.org/UpdateAction"}
}
```

Implementation

TripleWave is a **NodeJS** Web Application

NodeJS is a JavaScript runtime built on Chrome's V8 JavaScript engine.

- It uses an event-driven, non-blocking I/O model

Why NodeJS?

- It has very nice way to handle **data streams**

TripleWave is released with a Apache 2.0 Licence and the source code is hosted on github at:

- https://github.com/streamreasoning/TripleWave

Brief summary on NodeJS Stream

NodeJS provides three types of stream:

- ReadbleStream: stream that produce data
  - E.g., a file reader, a database connection, etc..
- WritableStream: stream that consume data
  - E.g., a file writer, an HTTP response, etc..
- TransformStream: stream that consume data, transform it and publish it
  - E.g., a JSON parser / serializer

Streams are EventEmitter

- You can attach EventListener to handle the emitted event

```
var stream; // some stream

stream.on('data',function(data){

  // do something

})
```

Brief summary on NodeJS Stream

NodeJS provides three types of stream:

- ReadbleStream: stream that produce data
  - E.g., a file reader, a database connection, etc..
- WritableStream: stream that consume data
  - E.g., a file writer, an HTTP response, etc..
- TransformStream: stream that consume data, transform it and publish it
  - E.g., a JSON parser / serializer

Streams workflows can be easily created by pipeing the streams toghether

```
var http = require('http');
var fs = require('fs');

var server = http.createServer(function (req, res) {
    var stream = fs.createReadStream(__dirname + '/data.txt');
    stream.pipe(res);
});
server.listen(8000);
```

https://github.com/substack/stream-handbook

Brief summary on NodeJS Stream (2)

How to create a custom stream (ECMAScript6):

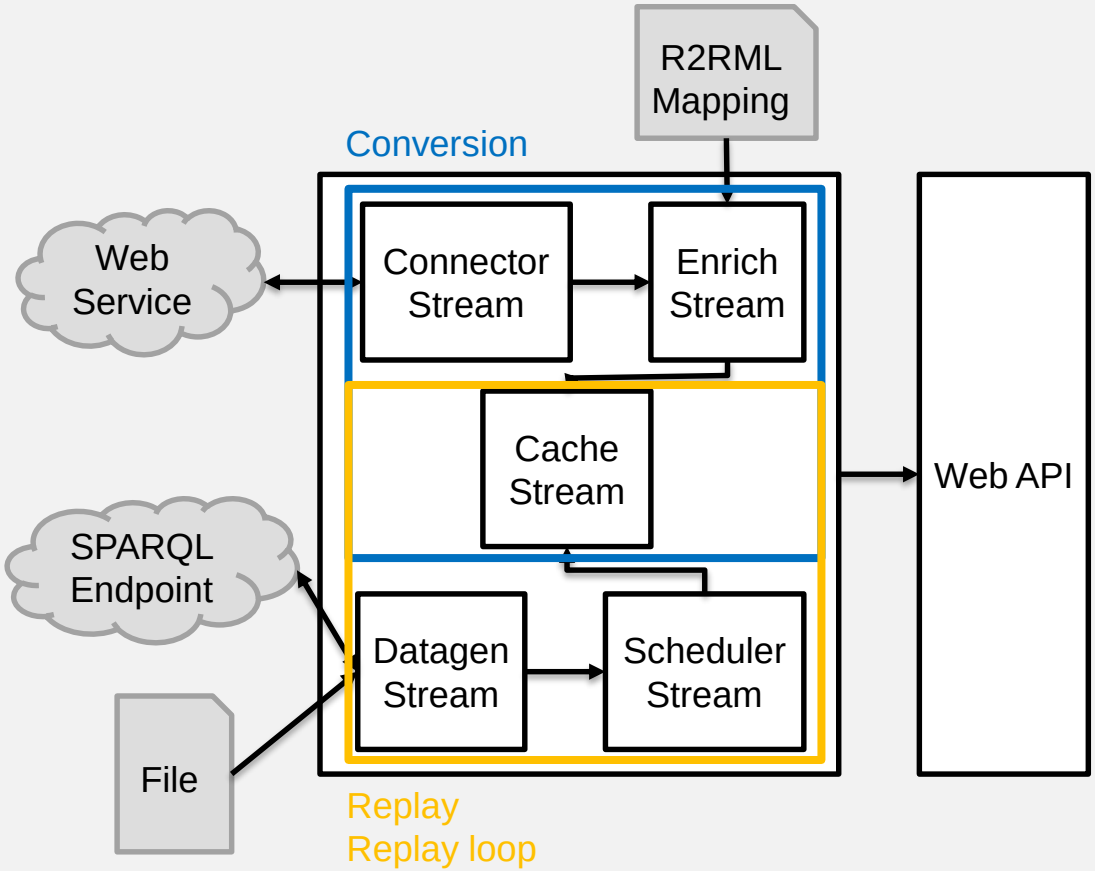
```
var stream = require('stream');

class Transformer extends stream.Transform {

  constructor(options) {
    super({
      readableObjectMode : true,
      writableObjectMode: true
    });
  }

  _transform(chunk, encoding, done) {
    this.push(chunk.email)
    done()
  }

}
```

Called every time the stream receive data

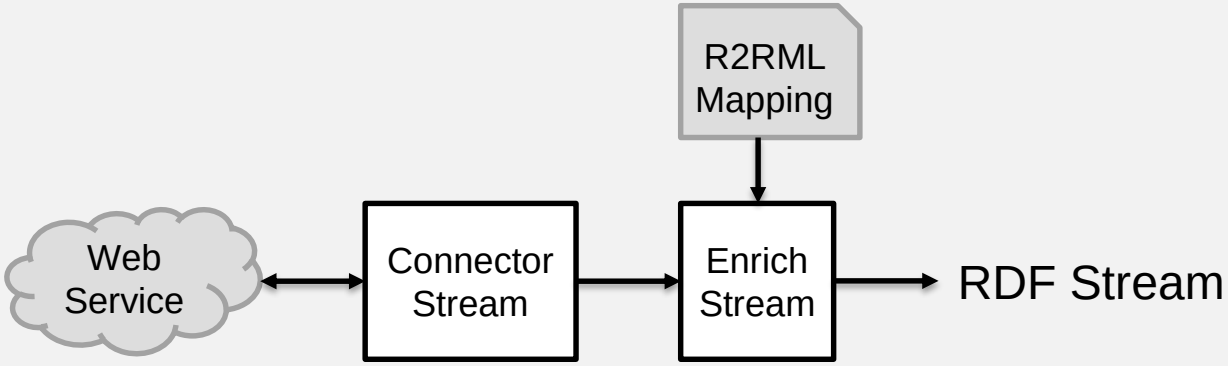
Push the data to the piped stream

https://gist.github.com/bhurlow/279243f279076c00f320

TripleWave Real* Architecture

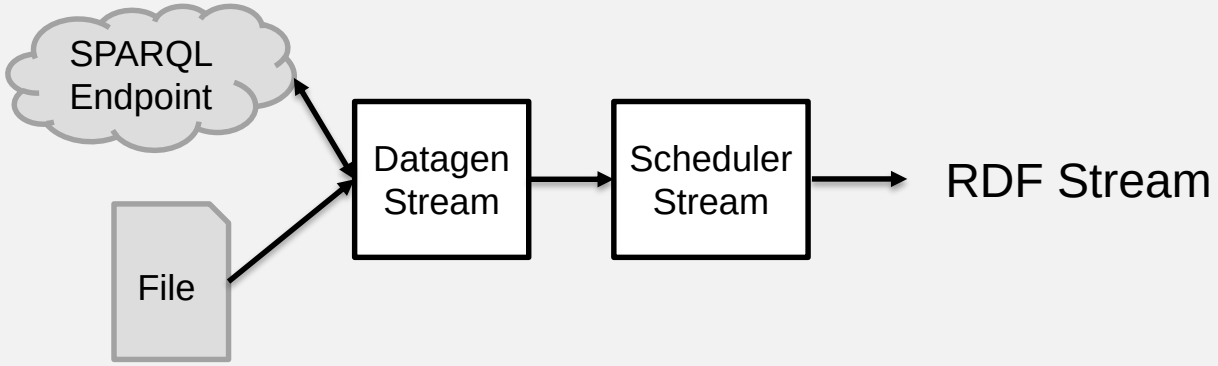


Conversion mode configuration



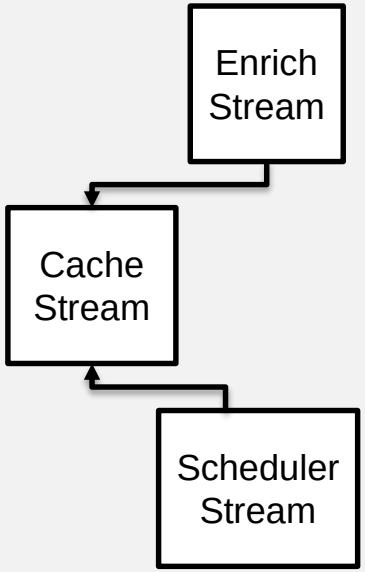
- Connector Stream: use the Web Service API to retreive data and publish them as a NodeJS Stream.
- Enrich Stream: loads the R2RML Mapping and applies the transformation to the data.

Replay mode configuration



- Datagen Stream: load the data from a SPARQL endpoint or from a file
- Scheduler Stream: read the timestamp and push forward the data accordingly

Cache Stream



- It caches the last 100 triples
- It provides methods to access the data

Web API

Allows the access to the data through HTTP or WebSocket

In particular:

- Retrieve the sgraph of the data and the last 100 cached elements
  - GET http://path_to_triplewave/sgraph
- Retrieve the details of a single triple
  - GET http://path_to_triplewave/:id

- Retreive the live stream through HTTP
  - GET http://path_to_triplewave/stream
- Retrive the live stream through WebSocket
  - GET ws://path_to_triplewave/primus

How to install

- Requirements
  - NodeJS >= v6.0.0
  - Java 8

- Clone the GitHub repository
  - git clone https://github.com/streamreasoning/TripleWave.git

- Install the dependency
  - npm install

How to run TripleWave

TripleWave can be fully customized with the **configuration file** found in the /config folder.

It also accepts **command line parameter**, and the overwrite the values present in the configuration file.

- -c, --configuration: path to a configuration file (/config/config.properties as default)
- -m, --mode: running mode (transform | replay | endless)
- -s, --sources: source of the data (triples | rdfstream)

To run simply launch ./start.sh

How to run TripleWave – Converting Wikipedia Changes Stream

On Linux/Mac: \start.sh –-mode transform

On window: node app.js --mode=transform

Data Structure:


```
{ channel: '#en.wikipedia',
 wikipedia: 'English Wikipedia',
 page: 'Persuasion (novel)',
 pageUrl: 'http://en.wikipedia.org/wiki/Persuasion_(novel)',
url: 'http://en.wikipedia.org/w/index.php?diff=498770193&oldid=497895763', delta: -13,
comment: '/* Main characters */',
wikipediaUrl: 'http://en.wikipedia.org',
user: '108.49.244.224',
userUrl: 'http://en.wikipedia.org/wiki/User:108.49.244.224',
unpatrolled: false,
newPage: false,
robot: false,
anonymous: true,
namespace: 'Article'
flag: '' }
```

How to run TripleWave - Replaying the Linked Sensor Data stream

On Linux/Mac: \start.sh –-mode endless|replay –-sources triples

On Window:

1. Start Fuseki with: java -jar fuseki\jena-fuseki-server-2.3.1.jar --update --mem \ds &
2. Node app.js --mode=endless|replay --sources=triples

In this case the script will also start Fuseki

- Then TripleWave will load the data and start the stream

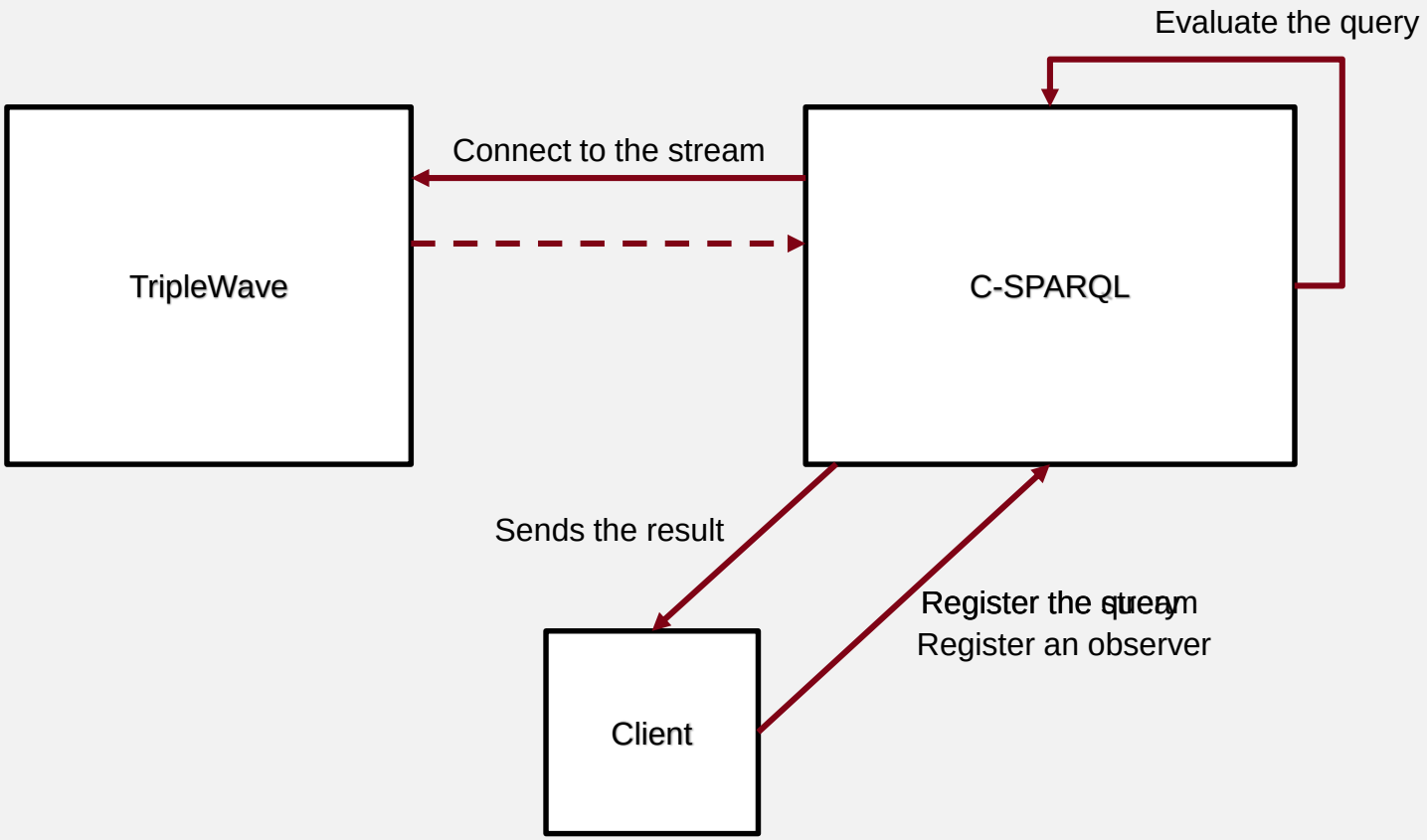
How to run TripleWave - Examples

Replaying the Social Graph Stream

On Linux/Mac: .\start.sh --mode endless|replay --sources rdfstream

On Window: node app.js --mode=endless|replay --sources rdfsteam

How to consume the TripleWave Stream



How to consume the TripleWave Stream with C-SPARQL

1. Register a new RDF stream into the C-SPARQL engine

   PUT http://localhost:8175/streams/<*stream_URI*>

2. Register a new query into the C-SPARQL engine

   PUT http://localhost:8175/queries/<query_name>

3. Add an Observer to a query to retrieve the results

   POST http://localhost:8175/queries/<query_name>

Let's see the demo

What we learnt so far

How to use TripleWave for publishing RDF Streams

- From Web sources
- From SPARQL endpoint
- From files

How TripleWave is able to create RDF stream from those sources

How to feed its stream in a RSP Service in order to perform queries over the data

Advertise time

Resource Paper presentation on Thursday

TripleWave: Spreading RDF Streams on the Web

13:30 - 14:50

Parallel Session 4

Streams Room 501

[R] Planning Ahead: Stream-Driven Linked-Data Access under Update-Budget Constraints — 13:30
Shen Gao, Daniele Dell'Aglio, Soheila Dehghanzadeh, Abraham Bernstein, Emanuele Della Valle and Alessandra Mileo

[R] SPARQL-to-SQL on Internet of Things Databases and Streams — 13:50
Eugene Siow, Thanassis Tiropanis and Wendy Hall

[J] Operator-aware approach for boosting performance in RDF stream processing — 14:10
Danh Le-Phuoc

[S] TripleWave: Spreading RDF Streams on the Web — 14:30
Andrea Mauri, Jean-Paul Calbimonte, Daniele Dell'Aglio, Marco Balduini, Marco Brambilla, Emanuele Della Valle and Karl Aberer

Tutorial on RDF Stream Processing 2016

M.I. Ali, J-P Calbimonte, D. Dell'Aglio,
E. Della Valle, and A. Mauri

http://streamreasoning.org/events/rsp2016

ISWC 2016
Kobe, Japan
The 15th International Semantic Web Conference

Insight

Hes·so VALAIS WALLIS

Universität Zürich[UZH]

POLITECNICO MILANO 1863

How to publish RDF Stream with TripleWave

Andrea Mauri

andrea.mauri@polimi.it