A Proposal for Publishing Data Streams as Linked Data

http://streamreasoning.org

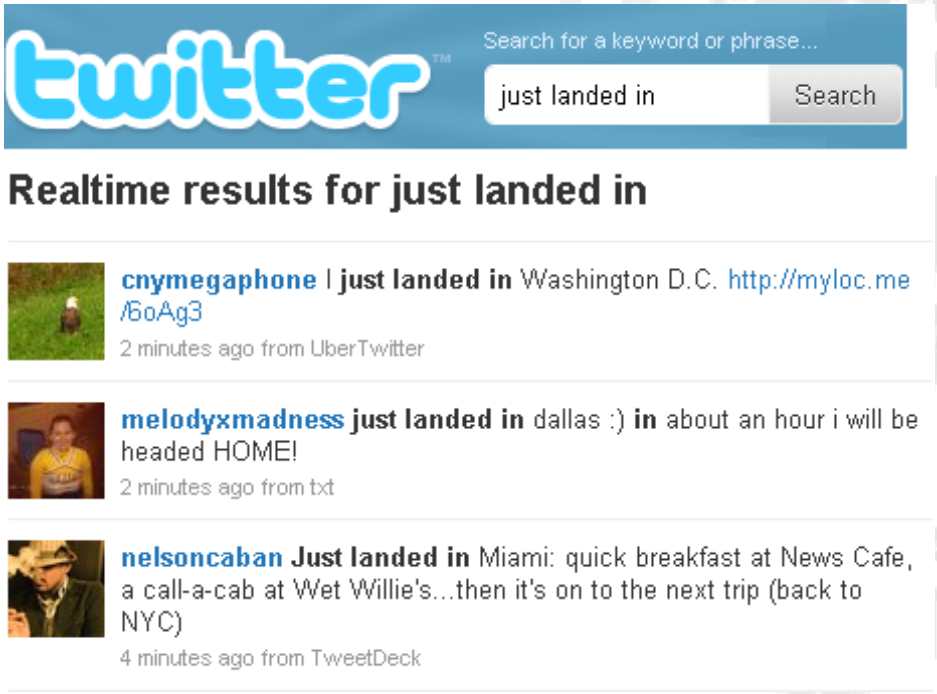
http://wiki.larkc.eu/c-sparql/

Davide F. Barbieri
DEI – Politecnico di Milano
dbarbieri@elet.polimi.it

Emanuele Della Valle
DEI – Politecnico di Milano
emanuele.dellavalle@polimi.it

Introduction
Real-Time Streams on the Web

- Streams are appearing more and more often on the Web in sites that distribute and present information in real-time streams.
- E.g. http://twitter.com/#search?q=just%20landed%20in



- Checkout http://activitystrea.ms/ for a standard API

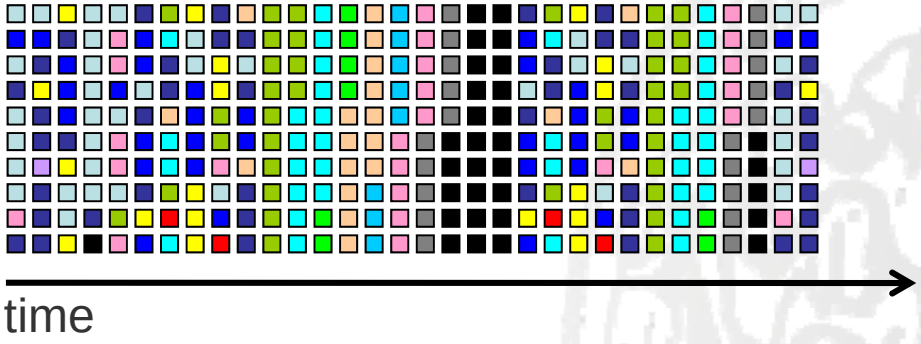
Introduction
Combining Streams and Static Information

- We anticipate a rapidly growing need of mashing up this streaming information with more static one.
- E.g., Twitter + MetaCarta

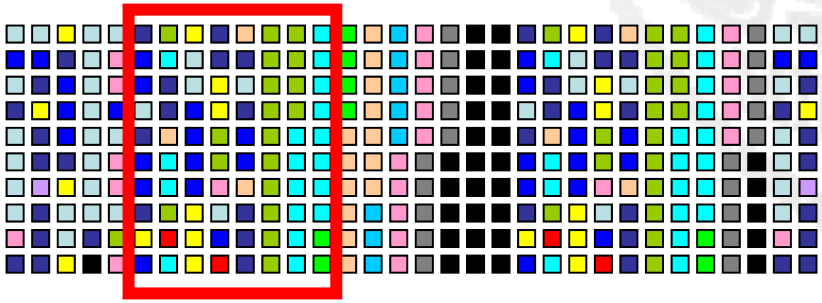
[source: http://blog.blprnt.com/blog/blprnt/just-landed-processing-twitter-metacarta-hidden-data]

Background
Managing Streams

- Streams
  - unbounded sequences of time-varying data elements



- Stream Processing
  - **Continuous queries registered** over streams that are observed trough **windows**



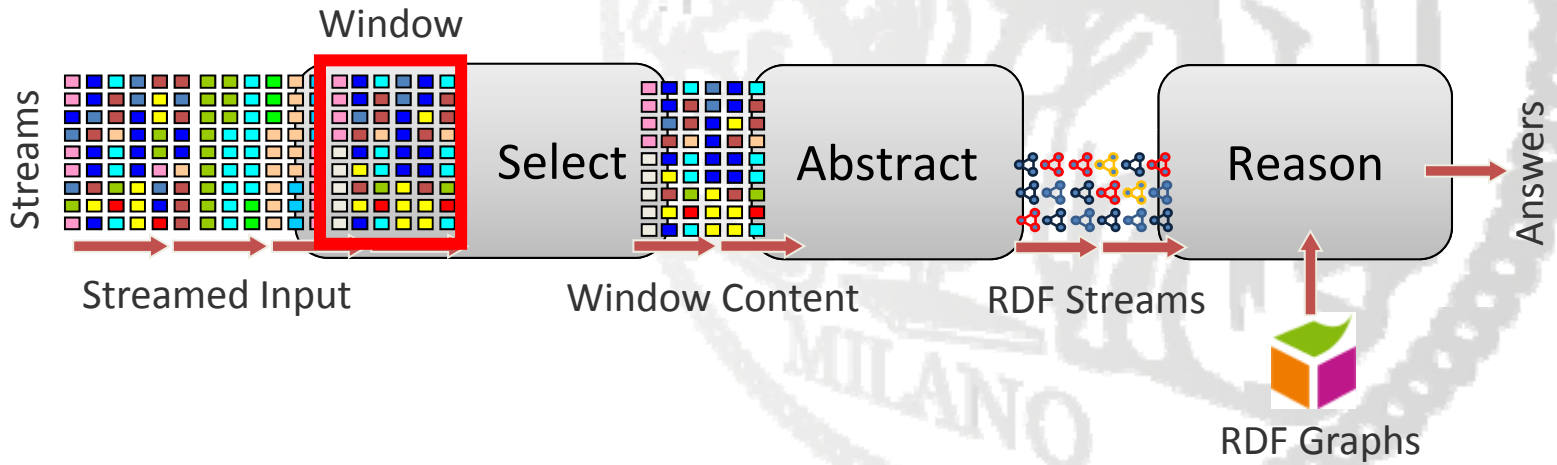
State-of-the-Art
Data Stream Management Systems (DSMS)

- Research prototypes:
  - STREAM http://infolab.stanford.edu/stream/
  - Aurora http://www.cs.brown.edu/research/aurora/
  - Borealis http://www.cs.brown.edu/research/borealis/public/
- Some Features are embedded in
  - Oracle http://www.oracle.com/technology/products/dataint/htdocs/streams_fo.html
  - DB2 http://www.eweek.com/c/a/Database/IBM-DB2-Turns-25-and-Prepares-for-New-Life/
- Start-ups
  - StreamBase http://www.streambase.com/
- Open Source
  - Esper http://esper.codehaus.org/
  - Data Turbine http://www.dataturbine.org/

Background
Continuous SPARQL (C-SPARQL)

- What is it?
  - an extension to SPARQL for continuous querying over (virtual) streams of RDF and static RDF graphs

- Architecture of our C-SPARQL Engine
  - Based on the Large Knowledge Collider (LarKC) conceptual framework



Background
RDF Stream

- RDF Stream Data Type
  - Ordered sequence of pairs, where each pair is made of an RDF triple and its timestamp `t`

    `(< triple >, t)`
- E.g.,

  `(< :traveller1 :justLanded :placeA >, T1)`

  `(< :traveller2 :justLanded :placeB >, T1)`

  `(< :traveller3 :justLanded :placeA >, T2)`

  `(< :traveller1 :justLanded :placeC >, T3)`

Background
An Example of C-SPARQL Query

Who has landed in USA in the last hour?

```
REGISTER QUERY WhoHasLandedInUSAinTheLastHour AS
PREFIX gno: <http://www.geonames.org/ontology#>
PREFIX c: <http://www.geonames.org/countries/#>
PREFIX : <http://example>
SELECT ?traveller ?place ?type
FROM <http://sws.geonames.org/nonExistingUSfeatureGraph>
FROM STREAM <http://someStreamGeneratedFromTwitter>
[ RANGE 60m STEP 5m ]
WHERE {
 ?traveller :justLanded ?place .
 ?place gno:inCountry c:US .
 ?place gno:featureCode ?type .
}
```

Background
An Example of C-SPARQL Query Explained

Who has landed in USA

Query registration (for continuous execution)

```
REGISTER QUERY WhoHasLandedInUSAinTheLastHour AS
PREFIX gno: <http://www.geonames.org/ontology#>
PREFIX c: <http://www.geonames.org/countries/#>
PREFIX : <http://example>
SELECT ?traveller ?place
FROM <http://sws.geonames.org/nonExistingUSfeatureGraph>
FROM STREAM <http://someStreamGeneratedFromTwitter>
[ RANGE 60m STEP 5m ]
WHERE {
 ?traveller :justLanded ?place .
 ?place gno:inCountry c:US .
 ?place gno:featureCode ?type .
}
```

FROM STREAM clause

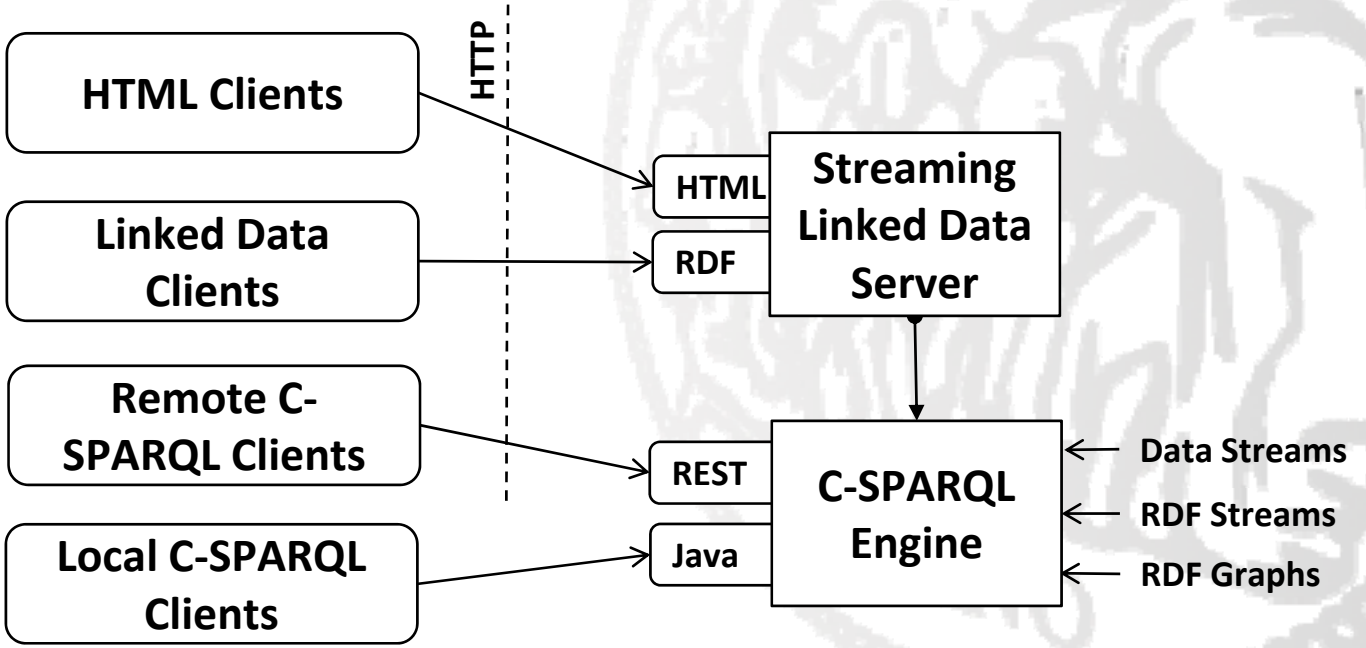
WINDOW

triples from a stream

Combined with triples a RDF graph

Proposal
Streaming Linked Data

- What
  - an extension of our C-SPARQL Engine that publishes data streams as Linked Data
- Architecture



Streaming Linked Data
Raw Data Stream

- The problem
  - How to publish as linked an RDF Stream?
- Proposal
  - Use Named Graph
  - A *Stream Graph* (s-graph)
    - a metadata graph that describes the current content of the window over the stream
  - Several *Instantaneous Graphs* (i-graph)
    - one for each time stamp
  - **`rdfs:seeAlso`** is used (and reserved) to link the graphs
  - A *Streaming Linked Data Vocabulary* is used to describe the content of the graphs

Streaming Linked Data
Raw Data Stream - Example

A s-Graph (only metadata) and …

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema #> .
@prefix sld: <http://www.streaminglinkeddata.org/schema#> .
@prefix : <http://example/> .
:sgraph  sld:lastUpdate "T3"^^xsd:dataTime;
         sld:expires    "T4"^^xsd:dataTime;
         sld:windowType sld:logicalTumbling;
         sld:windowSize "PT1H"^^xsd:duration .
:sgraph1 rdfs:seeAlso :igraph1 .
:igraph1 sld:receivedAt "T1"^^xsd:dataTime .
:sgraph1 rdfs:seeAlso :igraph2 .
:igraph2 sld:receivedAt "T2"^^xsd:dataTime .
:sgraph1 rdfs:seeAlso :igraph1 .
:igraph1 sld:receivedAt "T3"^^xsd:dataTime .
```

Streaming Linked Data
Raw Data Stream - Example

… and three i-Graphs (triples + few metadata)

```
:igraph1 sld:receivedAt "T1"^^xsd:dataTime ;
         rdfs:seeAlso :sgraph .
:traveller1 :justLanded :placeA .
:traveller2 :justLanded :placeB .

:igraph2 sld:receivedAt "T2"^^xsd:dataTime ;
         rdfs:seeAlso :sgraph .
:traveller3 :justLanded :placeA .

:igraph3 sld:receivedAt "T3"^^xsd:dataTime ;
         rdfs:seeAlso :sgraph .
:traveller1 :justLanded :placeC .
```

igraph1

igraph2

igraph3

Streaming Linked Data
Raw Data Stream – naming the graphs

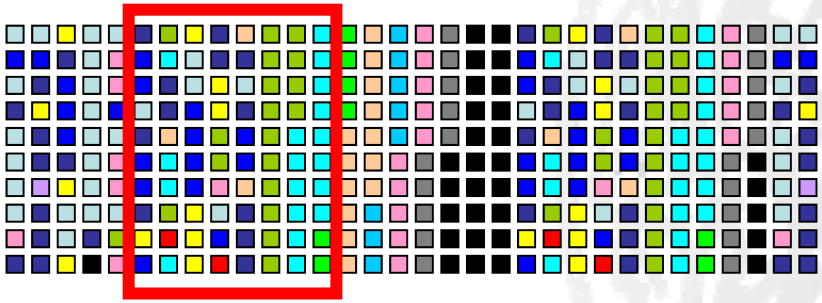
- Patterns
  - s-graphs `http://ex.org/%stream-name%`
  - i-graphs `http://ex.org/%stream-name/%timestamp%`
- Example
  - s-graph
    `http://ex.org/just-landed-in`
  - i-graphs
    `http://ex.org/just-landed-in/2010-02-12T133441Z`
    `http://ex.org/just-landed-in/2010-02-12T133710Z`
    `http://ex.org/just-landed-in/2010-02-12T133933Z`

Streaming Linked Data
Raw Data Stream – dereferencing rules

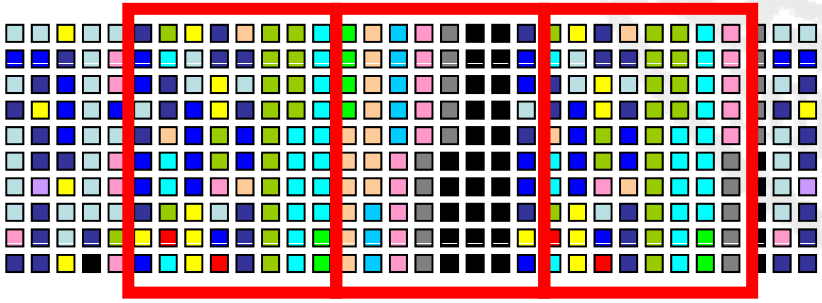
- Resource
  `http://ex.org/just-landed-in`
  `http://ex.org/just-landed-in/2010-02-12T133441Z`
- Linked Data Clients
  `http://ex.org/trdf/just-landed-in`
  `http://ex.org/trdf/just-landed-in/2010-02-12T133441Z`
- HTML Clients
  `http://ex.org/page/just-landed-in`
  `http://ex.org/page/just-landed-in/2010-02-12T133441Z`

Streaming Linked Data
Controlling the Window - Types of window

- **physical**: a given number of triples
- **logical:** a variable number of triples which occur during a given time interval (e.g., 1 hour)
  - **Sliding**: they are progressively advanced of a given STEP (e.g., 5 minutes)



  - **Tumbling**: they are advanced of exactly their time interval



Streaming Linked Data
Controlling the Window

- Physical Windows
  - Schema

    `http://ex.org/%stream-name%/`**`physical/%size%`**
  - Example, last 1000 triples

    `http://ex.org/just-landed-in/`**`physical/1000`**
- Logical Windows
  - Schema

    `http://ex.org/%stream-name%/`**`logical/%size%/%step%`**
  - Example, last hour sliding with a step of 1 minute

    `http://ex.org/just-landed-in/`**`logical/PT1H/PT10M`**

- NOTE 1: These **URL patters are translated in equivalent C-SPARQL queries** that select a part of the stream
- NOTE 2: The lexical space of an interval is the same as xsd:duration, i.e., the format **PnYnMnDTnHnMnS** dened by ISO 8601

Streaming Linked Data
REST APIs to Control C-SPARQL Queries

- Operations:
  - **register** a C-SPARQL query
    - C-SPARQL queries have to be **registered** in the C-SPARQL Engine
  - **start** the computation
    - Be aware that the results are not “semantically” valid until the window is completely filled in
  - **pause** the computation
    - the windowing mechanism will keep working, but the window content is not processed
  - **stop** the computation
    - the window is emptied
  - **unregister** the C-SPARQL query

Conclusion and Retrospective

- In our **previous work** we investigated **C-SPARQL** as an approach **to treat non-RDF DSMSs as virtual RDF streams** and graphs.
- With **this position paper**, we propose an extension of our C-SPARQL Engine that publishes data **streams as Linked Data**.
- We introduced the concept of
  - A ***Stream Graph*** (s-graph)
    - a metadata graph that describes the current content of the window over the stream
  - Several ***Instantaneous Graphs*** (i-graph)
    - one for each time stamp
  - **`rdfs:seeAlso`** is used (and reserved) to link the graphs
- We define URL patterns
  - to control the window
  - to register, start, pause, stop, unregister a C-SPARQL query

Much More to Come! Questions?

Thank You!
Keep an eye on
http://www.larkc.eu/
http://streamreasoning.org/

Emanuele Della Valle
DEI – Politecnico di Milano
emanuele.dellavalle@polimi.it