It’s a dynamic World!

Ubiquitous Streams and the Linked Data Web
1. What are we Facing?
2. What can we Solve?
3. Where do we Go?
The Ingredients
A connected digital layer!
There are economic Arguments
The Internet of Everything (IoE) creates $14.4 trillion in Value at Stake — the combination of increased revenues and lower costs that is created or will migrate among companies and industries from 2013 to 2022.

Cisco whitepaper, 2013
Many different Domains

Over 200 open data sets with more than 25 billion facts, interlinked by 400 million typed links, doubling every 10 month!

http://lod-cloud.net/

Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
... but there are more ...
Graphs will be the basic representation
Enormous amounts of DATA
“90% of the data in the world today has been created in the last two years alone” – IBM

“The bringing together of a vast amount of data from public and private sources [...] is what Big Data is all about” – IDC

Over the next few years we’ll see the adoption of scalable frameworks and platforms for handling streaming, or near real-time, analysis and processing.” – O’Reilly

Big Data represents a number of developments in technology that have been brewing for years and are coming to a boil. They include an explosion of data and new kinds of data, like from the Web and sensor streams; [...].” – IDC

Figure 3: By 2020, digital records will be 44 times larger than in 2009

Global digital data (in exabytes)

Source: IDC
You may say now:

“We know all that!”

(please not AGAIN)
But now – it’s DYNAMIC
Fasten seatbelts!
...and it **links** with static data
Every-THING / -BODY is a “sensor”
THIS CHANGES THE GAME
What can we SOLVE?
The Problem
The Goal
Technically speaking ...

- **Semantic descriptions** of sensors, streams, events, observations, etc.
- “Senso ergo sum” – semantic descriptions **down to the sensor level**
- **Web protocols** down to the **sensor level**
- **SPARQL-like access** to streams and sensors
- **Infrastructure** frameworks
- **Reasoning** with streams

... so that stream are **just another form** of Linked Data
Research Challenges

- Query processing model and execution framework
- Efficient physical organization of Linked Stream Data
- Efficient incremental evaluation algorithms for graph-based continuous query patterns
- How to dynamically optimize continuous queries over Linked Stream Data
Keep it small and simple!

Application := Data + Services

Application

LOD

REST
KISS extended

Application

CQELS

REST

SPARQL

Linked Streams

Virtual Sensors

COAP Sensors

Linked Data
Sensors, streams, events, observations
W3C SSN XG

- Semantic Sensor Networks ontology to describe sensors and sensor data

- Semantic annotations for OGC’s SWE Sensor Model Language

Location (as a Feature)

```xml
<sos:featureOfInterest
    xlink:href="http://sws.geonames.org/5248611/"
    xlink:role="http://purl.oclc.org/NET/ssnx/ssn#FeatureOfInterest"
    xlink:arcrole="http://www.w3.org/ns/sawsdl#modelReference"/>
```

- Motivations
  - No existing sensor ontology included all the basic concepts
  - Ease integration of (some) semantics in more widespread languages and standards (specifically SensorML)
Relation to existing standards

Sensor and Sensor Networks  Sensor Web  Semantic Web

Proprietary (or no) standards  OGC standards  W3C standards

Data-focused ops

Application services  Mashup services

Sensor and Sensor Network APIs

Devices (nodes & sensors)

Network-focused ops
Device-focused ops
SSN-XG Ontology Structure

- Deployment
- System
- OperatingRestriction
- PlatformSite
- Device
- Process
- Data
- Skeleton
- MeasuringCapability
- ConstraintBlock
SSN-XG Ontology Structure
SSN Application: SPITFIRE

Concepts on sensor network topology and devices

Sensor Datasets

Event Datasets

LOD Cloud

Concepts on sensor role, events, sensor project

- **DUL**: DOLCE+DnS Ultralite
- **EventF**: Event-Model F
- **SSN**: SSN-XG
- **CC**: Contextualised-Cognitive
SPITFIRE Vocabulary

http://www.spitfire-project.eu
Size matters!

- OS + 6LowPAN + CoAP + Semantic description < 48kB?
- Processing power?
## Storage requirements

<table>
<thead>
<tr>
<th>Triples</th>
<th>Turtle (req. bytes)</th>
<th>Ntriples (req. bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>587</td>
<td>74.479</td>
<td>111.751</td>
</tr>
<tr>
<td>195</td>
<td>33.152</td>
<td>60.400</td>
</tr>
<tr>
<td>117</td>
<td>17.054</td>
<td>27.699</td>
</tr>
<tr>
<td>92</td>
<td>10.248</td>
<td>12.416</td>
</tr>
<tr>
<td>83</td>
<td>9.452</td>
<td>11.064</td>
</tr>
<tr>
<td>76</td>
<td>8.891</td>
<td>10.225</td>
</tr>
<tr>
<td>65</td>
<td>7.641</td>
<td>8.720</td>
</tr>
<tr>
<td>53</td>
<td>6.406</td>
<td>7.179</td>
</tr>
<tr>
<td>40</td>
<td>5.035</td>
<td>5.520</td>
</tr>
<tr>
<td>34</td>
<td>4.390</td>
<td>4.681</td>
</tr>
<tr>
<td>30</td>
<td>3.919</td>
<td>4.108</td>
</tr>
<tr>
<td>27</td>
<td>3.565</td>
<td>3.677</td>
</tr>
<tr>
<td>23</td>
<td>3.091</td>
<td>3.100</td>
</tr>
<tr>
<td>22</td>
<td>2.971</td>
<td>2.976</td>
</tr>
<tr>
<td>21</td>
<td>2.852</td>
<td>2.810</td>
</tr>
<tr>
<td>16</td>
<td>2.339</td>
<td>2.170</td>
</tr>
<tr>
<td>10</td>
<td>1.679</td>
<td>1.356</td>
</tr>
<tr>
<td>8</td>
<td>1.470</td>
<td>1.097</td>
</tr>
<tr>
<td>4</td>
<td>999</td>
<td>523</td>
</tr>
<tr>
<td>3</td>
<td>898</td>
<td>396</td>
</tr>
<tr>
<td>2</td>
<td>798</td>
<td>271</td>
</tr>
<tr>
<td>1</td>
<td>701</td>
<td>149</td>
</tr>
</tbody>
</table>
We can describe Sensors and their data.
RESTful sensor interfaces

• Standardisation
  • Physical: 802.15.4
  • Network: IEEE 6LoWPAN, ROLL

• Service layer:
  • IETF CoRE (Constrained RESTful Environments): CoAP protocol + extensions
  • Encoding (Extensible XML interchange - EXI, SensorML)
  • Ontologies

• CoAP = Constrained Application Protocol
  • IETF draft, http://tools.ietf.org/id/coap
  • Core proposal + > 17 extensions
CoAP = HTTP for sensors
CoAP Example

- Accessing sensors from web browser using HTTP-CoAP proxying – SPITFIRE Smart Service Proxy (SSP)

- measurements; m=GET; d="Sensor measurements"
- .well-known/serverInfo; m=GET; d="AIAddressLength|AIAddress|NTI|NameType|Name"
- devInfo; m=GET; d="Device Info"
OK, we can access sensors via RESTful interfaces now.
KISS revisited

Application

CQELS → REST → SPARQL

GSN → Linked Streams

Virtual Sensors → COAP Sensors

Linked Data

Twitter, Phone, User, Data Cloud
Linked Stream Processing

A. Linked Open Data cloud

B. Query

```
SELECT ?person FROM ... [NOW]
WHERE {
  ?person ...
}
```

C. Linked Stream Data

pre-processing

optimization

execution

answer
Sensor metadata

weatherStation

aTemperature

dublinAirport

aHumidity

latestWeather

tempValue

humidValue

readings

stream data

ssn:observes

ssn:observedBy

ssn:isPropertyOf

ssn:observedProperty

ssn:featureOfInterest

ssn:hasValue

ssn:observationResult

ssm:observedProperty

ssm:value

ssm:unit

"18"^xsd:float

"Celcius"

"60"^xsd:float

"%"

Stream data snapshot at 2011-07-08T21:32:52
CQELS-QL query language

CQELS-QL – an extension to SPARQL 1.1

Construct new RDF stream

Stream pattern

CONSTRUCT {?person1 lv:reachable ?person2}
FROM NAMED <http://deri.org/floorplan/>
WHERE {
  STREAM <http://deri.org/streams/rfcid> [NOW] {?person1 lv:detelectedAt ?loc1}
  STREAM <http://deri.org/streams/rfcid> [RANGE 3s] {?person2 lv:detelectedAt ?loc2}
  GRAPH <http://deri.org/floorplan/> {?loc1 lv:connected ?loc2} }

SELECT ?coAuthName
FROM NAMED <http://deri.org/floorplan/>
WHERE {
  STREAM <http://deri.org/streams/rfcid> [TRIPLES 1] {?auth lv:detelectedAt ?loc}
  STREAM <http://deri.org/streams/rfcid> [RANGE 5s] {?coAuth lv:detelectedAt ?loc}
    ?auth foaf:name `$Name$`. ?coAuth foaf:name ?coAuthorName} }
 FILTER (?auth != ?coAuth) }
Processing Model: Operators

- **Triple-based window operators** extracts triples from RDF stream or dataset that
  - match a given triple pattern
  - valid within in a time window

- **Relational operators** enable employing relational algebras in the processing model

- **Streaming operators** generate new streams from output of other operators based on graph templates
Black Box Approach

Query

Optimizer

Executor

Operator implementations

Execution

Access methods

Query rewriter

Orchestrator

Overhead

Data transformation

SPARQL-like
White Box Approach

"SPARQL" Query

Logical Query Plan

Optimal Plan based on operator impl.

cost > max_cost ⇒ re-optimize

Physical Operators

B+Tree hash table triple-based indices

Native Access methods

RDF dataset

Linked datasetstream

Physical Access (data structures)

Nested loop or hash joins aggregation, sliding windows

Operator implementations

Execution

f(x)

Optimizer

Executor

Query

Optimal Plan based on operator impl.
CQELS Architecture

Latest DB and stream technology extended and optimized for Linked Data (science is a “toolbox”)
CQELS + EC2 + Storm

https://code.google.com/p/cqels/wiki/CQELSCloud
Updates /sec
Lookups / sec
Window size
Parallel queries
Performance

- 33 million triples
- 10000 parallel queries
- 32 EC2 nodes (medium instance, 3.5GB, 1 virtual core with 2 EC2 Compute)

CQELS cloud configuration:
- 32 nodes Hadoop + HBase region server + storm supervisor
- 1 Nimbus node (Storm master), 2 Zookeeper nodes, 1 Hbase master node

- Lookup: 23881 inputs/sec
- Max window size: 100000 patterns
- Matching operator: 339978 inputs/sec
- Dictionary insert: 65000 inputs/sec
## Scalability: Triple/sec

<table>
<thead>
<tr>
<th>EC2 nodes</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary join</td>
<td>2897</td>
<td>4420</td>
<td>9638</td>
<td>11696</td>
<td>23881</td>
</tr>
<tr>
<td>Match(100k)</td>
<td>2011</td>
<td>6003</td>
<td>17899</td>
<td>31996</td>
<td>43117</td>
</tr>
<tr>
<td>Match(10k)</td>
<td>48143</td>
<td>81000</td>
<td>98132</td>
<td>163111</td>
<td>339978</td>
</tr>
<tr>
<td>5-way join</td>
<td>16978</td>
<td>19982</td>
<td>21997</td>
<td>40193</td>
<td>60197</td>
</tr>
<tr>
<td>Aggregation</td>
<td>52381</td>
<td>76132</td>
<td>125314</td>
<td>149978</td>
<td>220165</td>
</tr>
</tbody>
</table>
More Tests required
## Functionality Tests

### No engine supports all SPARQL query patterns

<table>
<thead>
<tr>
<th>Patterns covered</th>
<th>S</th>
<th>$N_P$</th>
<th>$N_S$</th>
<th>Engines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>J</td>
<td>A</td>
<td>E</td>
</tr>
<tr>
<td>$Q_1$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q_2$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q_3$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_4$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_5$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_6$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Patterns covered</th>
<th>S</th>
<th>$N_P$</th>
<th>$N_S$</th>
<th>Engines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>J</td>
<td>A</td>
<td>E</td>
</tr>
<tr>
<td>$Q_7$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>$Q_8$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q_9$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_{10}$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q_{11}$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$Q_{12}$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**F**: filter  
**J**: join  
**E**: nested query  
**N**: negation  
**T**: top k  
**U**: union  
**A**: aggregation  
**S**: uses static data  

$N_P$: number of patterns, $N_S$: number of streams, ⚫: syntax error, E: error, Ø: return no answer, X: not supported  

CQ: CQELS, CS: C-SPARQL, JT: JTALIS
Correctness Tests

<table>
<thead>
<tr>
<th></th>
<th>Rate: 100 inputs/sec</th>
<th>Rate: 1000 inputs/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CQELS</td>
<td>C-SPARQL</td>
</tr>
<tr>
<td>Q1</td>
<td>68</td>
<td>604</td>
</tr>
<tr>
<td>Q2</td>
<td>68</td>
<td>124</td>
</tr>
<tr>
<td>Q3</td>
<td>533</td>
<td>1065</td>
</tr>
<tr>
<td>Q4</td>
<td>11984</td>
<td>125910</td>
</tr>
<tr>
<td>Q10</td>
<td>28021</td>
<td>205986</td>
</tr>
</tbody>
</table>

Comparisons of I/O throughputs are invalid
Nevertheless ...

Application := Data + Services
... but it’s still too hard to use for me!
Simplify!

Application

Middleware

SQL2Q5

REST

COAP

Linked Streams

Virtual Sensors

COAP Sensors

Linked Data

Twitter

Cell Phone

Linking Data

Complex Data
Linked Stream Middleware
LSM: Live flights info

http://lsm.deri.ie/
LSM: Live train info

http://lsm.deri.ie/
LSM: Live traffic info

http://lsm.deri.ie/
Super Stream Collider

• The SSC is a platform provides a web-based interface and tools for building sophisticated mash-ups combining semantically annotated Linked Stream and Linked Data sources into easy to use resources for applications.
  • Drag & drop editor
  • CQELS/SPARQL visual editor
  • WebSocket stream publisher

http://superstreamcollider.org/
SSC Development Tools

http://superstreamcollider.org/
SSC Development Tools
SSC Development Tools

PREFIX lv: <http://deri.org/floorplan/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?locName
FROM NAMED <http://deri.org/floorplan/>
WHERE {
  STREAM <http://deri.org/states/rfid> [NOW]
  {?person lv:detectedAt ?loc}
  GRAPH <http://deri.org/floorplan/>
  {?loc lv:name ?locName}
  {?person foaf:name "AUTHORNAME"^^<http://www.w3.org/2001/XMLSchema#string> }
}
SSC Development Tools

http://superstreamcollider.org/
SSC Development Tools

http://superstreamcollider.org/
Awesome – I should try it for my next app!
Where do we Go?
Scalability
Stream Reasoning

StreamRule

Linked Data

Joint work with

Universität Potsdam

Stream ASP Logic Program

Answer Sets

Stream ASP Engine (oClingo)

Facts from query results

Continuous Query Engine (CQUELs)

Data Streams
Unfortunately, the world is uncertain, fuzzy, contradicting.
Combine Statistics & Logics
What about PRIVACY?
Build systems!
Don’t get hung up on labels!
ACKNOWLEDGEMENTS

DERI Team  Sponsors  Projects
Conclusions?