Using RDF to Describe Networks

Jeroen van der Ham
vdham@science.uva.nl

Advanced Internet Research Group
Universiteit van Amsterdam

September 30, 2005
Global Lambda Integrated Facility
What is GLIF?

- A group of cooperating NRENs, consortia and institutions
- Make lambdas available as integrated global facility
- Sharing their research and knowledge
- Each with different policies
Why Do We Need Network Descriptions?

- To provide an overview of resources
- Make path discovery easier
- Do simple problem detection
Why Do We Need Network Descriptions?

- To provide an overview of resources
- Make path discovery easier
- Do simple problem detection
Why Do We Need Network Descriptions?

- To provide an overview of resources
- Make path discovery easier
- Do simple problem detection
Why Do We Need Network Descriptions NOW?
Problem With Descriptions

- We need a description readable by both **humans** and **computers**.

- **Problem**: Computers still have no common sense.

**Example**

- ‘A is connected to B.’
- ‘There is a connection between A and B.’
Problem With Descriptions

- We need a description readable by both *humans* and *computers*.

- **Problem**: Computers still have no common sense.

**Example**

- ‘*A* is connected to *B*.’
- ‘There is a connection between *A* and *B*.’
Solution: Use Semantic Web techniques:

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.”

(Tim Berners-Lee)
Resource Description Format (RDF) is a Semantic Web technique.

- RDF is a lightweight ontology system
- It describes things using triplets:

Example:

- Document 1
  - author: "Jeroen van der Ham"
Subject, Property, Object

- Triplets consist of three elements:

**Definition**

- **Subject**: The thing it describes.
- **Predicate**: A property the statement describes.
- **Object**: The value of the property.
**RDF Data Model**

- RDF describes things using triplets:

**Example**

```
Document 1

author

Jeroen

name
affiliation
email

"Jeroen van der Ham"
"UvA"
"vdham@science.uva.nl"
```
RDF Data Model

RDF describes things using triplets:

- **author**
- **name**
- **affiliation**
- **email**

**Document 1**

- **Jeroen van der Ham**
- **name**: Jeroen
- **affiliation**: "UvA"
- **email**: "vdham@science.uva.nl"
RDF Namespaces

- Unique terminology is achieved using namespaces
- Readable for both computers and people

Example

- Uses **Dublin Core** Namespace.
Real-World RDF Examples

- **RSS**
  - RDF Site Summary (v1.0)
  - DOAP
    - Description of a Project
  - FOAF
    - Friend of a Friend
Real-World RDF Examples

- RSS
- RDF Site
- DOAP
- Description of a Project
- FOAF
- Friend of a Friend

SimpleParse

Description

SimpleParse is a BSD-licensed Python package providing a simple parser generator for use with the mxTextTools text-tagging engine. SimpleParse allows you to generate tagging tables for use with the text-tagging engine directly from your EBNF grammar. Unlike most parser generators, SimpleParse generates single-pass parsers (there is no distinct tokenization stage), an approach taken from the predecessor project (mxpars) which attempted to create "autonomously parsing regex objects." The resulting parsers are not as generalized as those created by, for instance, the Earley algorithm, but they do tend to be useful for the parsing of computer file formats and the like (as distinct from natural language and similar "hard" parsing problems). In addition to the parser generator, the SimpleParse project includes a sub-project to create a modified version of the mxTextTools engine which reorganizes the code to allow for certain common EBNF constructs.
Real-World RDF Examples

RSS RDF Site Summary (v1.0)

DOAP Description of a Project

FOAF Friend of a Friend
We started on a set of properties and classes to describe networks:

- Location
- Device
- Interface
Network Description Language

We started on a set of properties and classes to describe networks:

- **Location**
  - `locatedAt`
- **Device**
  - `hasInterface`
  - `name`
- **Interface**
  - `connectedTo`
  - `switchedTo`
  - `description`
Example of NDL

<ndl:Device rdf:about="#Rembrandt3">
  <ndl:name>Rembrandt3</ndl:name>
  <ndl:locatedAt rdf:resource="#Lighthouse"/>
  <ndl:hasInterface rdf:resource="#Rembrandt3:eth0"/>
</ndl:Device>
Example of NDL

```xml
<ndl:Device rdf:about="#Rembrandt3">
  <ndl:name>Rembrandt3</ndl:name>
  <ndl:locatedAt rdf:resource="#Lighthouse"/>
  <ndl:hasInterface rdf:resource="#Rembrandt3:eth0"/>
</ndl:Device>

<ndl:Interface rdf:about="#Rembrandt3:eth0">
  <ndl:name>Rembrandt3:eth0</ndl:name>
  <ndl:connectedTo rdf:resource="#Glimmerglass:port3"/>
</ndl:Interface>
```
Example of NDL

```xml
<ndl:Device rdf:about="#Rembrandt3">
  <ndl:name>Rembrandt3</ndl:name>
  <ndl:locatedAt rdf:resource="#Lighthouse"/>
  <ndl:hasInterface rdf:resource="#Rembrandt3:eth0"/>
</ndl:Device>

<ndl:Interface rdf:about="#Rembrandt3:eth0">
  <ndl:name>Rembrandt3:eth0</ndl:name>
  <ndl:connectedTo rdf:resource="#Glimmerglass:port3"/>
</ndl:Interface>

<ndl:Interface rdf:about="#Glimmerglass:port3">
  <ndl:name>Glimmerglass:port3</ndl:name>
  <ndl:connectedTo rdf:resource="#Rembrandt3:eth0"/>
</ndl:Interface>
```
SPARQL\(^1\) is a SQL-like query language for RDF:

**Example**

```
SELECT ?host1 ?host2
WHERE { ?if1 ndl:connectedTo ?if2 .
    ?if2 ndl:connectedTo ?if1 .
    ?host1 ndl:hasInterface ?if1 .
    ?host2 ndl:hasInterface ?if2 }
```

---

\(^1\)SPARQL Protocol And RDF Query Language
Query Result

Rembrandt3

Rembrandt4

Glimmerglass
<ndl:Interface rdf:about="#Glimmerglass:port3">
  <ndl:name>Glimmerglass:port3</ndl:name>
  <ndl:switchedTo rdf:resource="#Glimmerglass:port4"/>
</ndl:Interface>
Distributed Repositories

NDL descriptions can point to other network descriptions:

```xml
<ndl:Interface rdf:about="#Rembrandt3:eth0">
  <ndl:name>Rembrandt3:eth0</ndl:name>
  <ndl:connectedTo rdf:resource="nl:C6509:port7"/>
</ndl:Interface>

<ndl:Interface rdf:about="nl:C6509:port7">
  <rdfs:seeAlso rdf:resource="http://www.netherlight.nl/ndl.rdf"/>
</ndl:Interface>
```
Querying remote information can be done in two ways:

1. Fetch each description, parse it and then query it.
2. Issue SPARQL queries over HTTP or SOAP (WSDL Interface).

Both approaches allow for filtering.
Applying Network Descriptions in GLIF

1. Each institute creates a description of their network
2. Validate and publish using the portal
3. Description must be automatically updated with each change
4. Users use the portal or tools to find paths and resources
Extending NDL

RDF allows for easy extensibility:

- Include geographical information (geo) and use with Google Earth
- Link to FOAF descriptions of administrators
- Include policy information
- etc...
Future Research

- Publish a portal with links to participants
- Tools for automatic generation and updating of descriptions
- Extend NDL to include higher layers
- Security & filtering possibilities
Related Work

- Using RDF for Home Network Configuration
  *G. Klyne*

- Using the Semantic Web to Automate the Operation of a Hybrid Internetwork
  *Franco Travostino* (to be published, GridNets ’05)
Questions?

More information:
http://www.science.uva.nl/~vdham/research/ndl/