Using RDF to Describe Networks

Jeroen J. van der Ham [a,*] Freek Dijkstra [a] Franco Travostino [b]
Hubertus M.A. Andree [a] Cees T.A.M. de Laat [a]

[a] *Advanced Internet Research Group, Universiteit van Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, NL*

[b] *Nortel Labs, Boston MA, USA*

Abstract

Conventions such as iGrid 2005 and SuperComputing show that there is an increasing demand for more service options on networks. For such networks, large teams of experts are needed to configure and manage them. In order to make the full potential of hybrid networks available to the ordinary user, the complexity must be reduced.

This paper presents the idea of the Network Description Language (NDL), which builds on Semantic Web techniques to create a distributed Topology Knowledge Base (TKB). The TKB can provide a collection of reachability graphs, showing connectivity rules among physical and/or virtual entities.

Latching onto the Semantic Web provides network management with a new breed of tools—bots, compilers, browsers, both commercial off-the-shelf (COTS) and open source. The approach appears to be applicable to the Global Lambda Integrated Facility (GLIF) as well as other experimental communities.

Key words: Network Descriptions, Semantic Web, Resource Description Framework, Hybrid Networks

1 Introduction

A hybrid network is a challenging engineering feat: they offer end users both traditional IP services and new optical services in the form of dedicated lightpaths. The main purpose of hybrid networks is to efficiently and effectively navigate through the service options offered by Layers 3, 2, and 1. In such a

* Corresponding author.
Email address: vdham@science.uva.nl (Jeroen J. van der Ham).

network, islands that peer at Layer 3 will advertise reachability information along with service options that map to DiffServ Code Points[1]. Other islands that operate at Layer 2 will need to indicate, for instance, which one of the 10 Gigabit Ethernet framing techniques they use (e.g., LAN-/WAN PHY). And islands that operate at Layer 1 add a host of TDM and/or WDM options.

Enter Grids. Grids make a strong case for specialisation around a plurality of network services. Regular network traffic is typically many-to-many and is optimally handled within a packet routed network. Conversely, the super-sized traffic is often few-to-few and is best handled via agile Layer 1 services, as implemented in SURFnet[2], the research network of the Netherlands. The latter services yield a circuit-switched network experience with uncontested bandwidth and strong non-interference properties. A desirable Grid Network is therefore a network wherein packet routed segments and Layer 1 and 2 "cut-through" segments across different clouds can be assembled end-to-end by a discerning user or, better yet, a software agent on his behalf. The degree of user control is a sharp departure from inter-carriers or carrier-of-carriers agreements oblivious to specific traffic demands. In a Grid Network, a user (or software in his behalf) is seen playing an active role in the negotiation of service stipulations.

This paper presents the Network Description Language (NDL), which builds upon the Resource Description Format (RDF)[3] and its linking capabilities to produce a distributed Topology Knowledge Base (TKB)[4]. The advantage of the TKB is that it provides easily accessible knowledge that the management and control planes can build upon. This is especially true for the dynamic provisioning planes, such as UCLP[5], or DRAC[6].

This paper continues with a short introduction to the technology NDL builds upon, RDF and the Semantic Web, in section 2. Then in section 3 the Network Description Language is introduced. An overview of future research is given in section 4 and the conclusion is in section 5.

2 Introduction to RDF and the Semantic Web

The World Wide Web allows us to publish and share documents with other people in the world. However, because it has become so popular and so widespread, it has almost become the victim of its own success. The large scale and the abundant availability of data make it very hard to find what we want. Search-machines, such as Google or Yahoo have indexed the data and provide searching services. However, computers still have no common sense, so the search capabilities of the search machines are rather limited. Consider for example the following two sentences:

- A is connected to B.
- There is a connection between A and B.

These two sentences may have a different meaning. There is no way for a computer understands that these two lines have the same meaning. This is where the Semantic Web can help computers (and people). The following is an excerpt of the activity statement of the Semantic Web initiative[7]:

> *The goal of the Semantic Web initiative is to create a universal medium for the exchange of data where data can be shared and processed by automated tools as well as by people. For the Web to scale, tomorrow's programs must be able to share and process data even when these programs have been designed totally independently.*

In 2000 the Semantic Web initiative was started by the World Wide Web Consortium (W3C). Since then the W3C has been working on several specifications to publish and share (meta)data, including the Resource Description Framework (RDF)[3] and SPARQL[8]. In the following section we provide a brief introduction to RDF. Examples of RDF and SPARQL are given later on in section 3.

2.1 Resource Description Framework

The Resource Description Framework is a method for representing information about resources on the Web. It provides a common framework for expressing metadata so that it can be exchanged between applications without loss of meaning.

Information in RDF is expressed using triplets, with the following elements:

Subject The resource being described
Property The property the statement describes
Object The value of the property according to this statement

A set of triplets is called a graph. Using the property that an object can also be the subject of another triplet, complex graphs can be created. An example of such a graph is shown in figure 1. This graph shows that `Document1` is written by `Jeroen`. It also provides additional information about him, such as his name, affiliation and email address.

The graph in figure 1 has the same problem as the two lines shown before. We have provided an abstract way of defining relations, but use plain English as labels for identifying these relations. Consider the `author` relationship, we could have expressed this as `creator` without loss of meaning to human read-

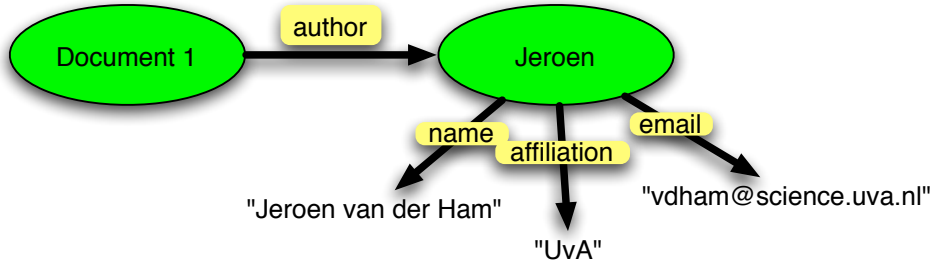


Fig. 1. A simple RDF graph

ers. RDF solves this terminology problem by using URIs [1] . Related terms are usually defined using the same URI-prefix, taking the form of XML namespaces. See for example the Dublin Core Metadata Initiative[10].

There are several ways of expressing RDF graphs. One is the graphical form as in figure 1. The most common textual form is *RDF/XML*[11], where the graph is encoded in an XML format. Another, more compact format is *Notation3*[12]. Throughout this paper we use the RDF/XML notation, which allows us to use tools for XML as well as RDF. Examples and explanation of the RDF/XML syntax are given in the next section.

An application of RDF is the Friend of a Friend (FOAF)[13] project. FOAF is similar to what we are trying to achieve with describing networks. That is, a distributed network of descriptions, managed by the responsible stake-holders, that together form a complex, navigable network. Tools are available to create FOAF descriptions[14], or to browse the network created by the links in the FOAF descriptions[15].

3 Network Description Language

The distributed descriptions and emergent network properties of the FOAF project are what we need for network descriptions. This is why we set out to create a simple ontology in RDF to describe physical networks, the Network Description Language. An overview of the classes and properties of the Network Description Language is given in figure 2.

NDL has three classes, shown at the top, which define the kind of resources:

Location Places where devices are located.
Device Physical devices that are part of a network.

[1] URI stands for Uniform Resource Identifier. A URL is one kind of URI. See also [9].

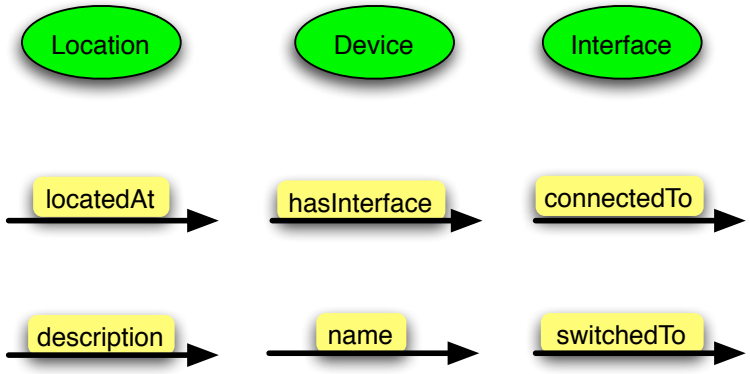


Fig. 2. Overview of the classes and properties of the Network Description Language

Interface The interfaces with which devices are connected to a network.

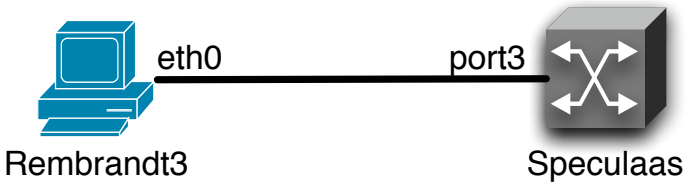
NDL has six properties, shown at the bottom in the figure, to define the relations between instances of the classes and other information.

locatedAt A relation between resources and their location.
hasInterface A relation between devices and interfaces.
connectedTo A relation between two interfaces, describing that they are *externally* connected.
description A relation that can be used to include a (human-readable) description of a resource.
name A relation between a resource and its name.
switchedTo This property is similar to *connectedTo*, but is used to describe internal connections, for example in optical cross-connects.

Using this limited set of simple classes and properties, it becomes possible to create descriptions of physical networks. An example of such a description is shown in figure 3. The picture at the top in figure 3 shows what is being described below it. We now explain how this is described using the Network Description Language.

Lines 7 to 11 of figure 3 define the device `Rembrandt3`. The `#`-prefix on line 7 states that the device is defined in the local namespace. Line 8 provides a human readable name and line 9 states that this device is located in the location `Lighthouse` (defined on lines 4 to 6). Finally, line 10 defines that `Rembrandt3` has an interface, `Rembrandt3:eth0`. This interface is defined on lines 12 to 15. The connection to another interface is defined on line 14, in this case `Speculaas:port3`. The `Speculaas` device (an optical switch) is defined similarly on lines 17–25.

Note that the connection between the `Rembrandt3` and the `Speculaas` is defined in both directions. This is used to denote that both devices are configured




```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:ndl="http://www.science.uva.nl/research/air/ndl#">
  <ndl:Location rdf:about="#Lighthouse">
    <ndl:name>Lighthouse</ndl:name>
  </ndl:Location>
  <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="#Speculaas:port3"/>
  </ndl:Interface>

  <ndl:Device rdf:about="#Speculaas">
      <ndl:name>Speculaas</ndl:name>
      <ndl:locatedAt rdf:resource="#Lighthouse"/>
      <ndl:hasInterface rdf:resource="#Speculaas:port3"/>
  </ndl:Device>
  <ndl:Interface rdf:about="#Speculaas:port3">
      <ndl:name>Speculaas:port3</ndl:name>
      <ndl:connectedTo rdf:resource="#Rembrandt3:eth0"/>
  </ndl:Interface>
</rdf:RDF>
```

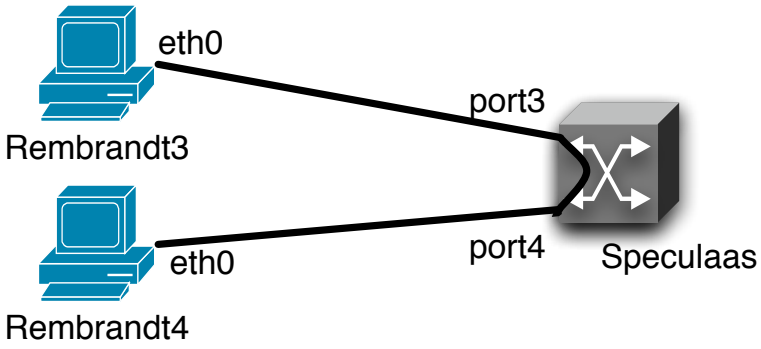
Fig. 3. An example description of a simple network.

properly and ensures the consistency of the description.

We can extend the description in figure 3 with another device, e.g. `Rembrandt4`, with a similar definition as `Rembrandt3`. This situation is depicted at the top in figure 4. To have an actual connection between `Rembrandt3` and `Rembrandt4`, we must define an *internal* connection using the `switchedTo` statement as shown in line 4. Just like the `connectedTo` statement, the `switchedTo` statement must be defined in both directions. The inverse definition for `Speculaas:port4` must also be given.

With the `connectedTo` and `switchedTo` statements as given above, we have defined a complete path from the device `Rembrandt3` to `Rembrandt4`. The difference between the `switchedTo` and the other properties is that `switchedTo` describes the (dynamic) state of the network [2].

[2] The `switchedTo` property is introduced here to demonstrate that NDL can also be used to describe the state of the network and is the subject of future research.




```
1  <ndl:Interface rdf:about="#Speculaas:port4">
2      <ndl:name>Speculaas:port4</ndl:name>
3      <ndl:connectedTo rdf:resource="#Rembrandt4:eth0"
4      <ndl:switchedTo rdf:resource="#Speculaas:port3"/>
5  </ndl:Interface>
```

Fig. 4. Example of a `switchedTo` statement.

3.1 Extracting Information from Network Descriptions

In the previous section we have shown how to describe networks using the Network Description Language. The examples show complete descriptions of the network, but usually only parts of the description is needed. For this purpose the W3C has created a query language for RDF: SPARQL.

SPARQL is an SQL-like query language for RDF. It uses a simple syntax to specify variables and triplet templates for retrieving information from an RDF store. An example is shown in figure 5.

```
1  SELECT  ?host1 ?host2
2  WHERE   { ?if1  ndl:connectedTo ?if2 .
3            ?if2  ndl:connectedTo ?if1 .
4            ?host1 ndl:hasInterface ?if1 .
5            ?host2 ndl:hasInterface ?if2 }
```

Fig. 5. Example of a SPARQL query.

The example shows a query to select two hosts, the variables `?host1` and `?host2`, which satisfy the constraints given in the `WHERE` clause (lines 2–5). The constraints use two additional variables `?if1` and `?if2`. These `if` variables must be `connectedTo` each other as specified by the first two lines. These interfaces must also belong to the `host`s, as specified by the last two lines. This query will return all host pairs that are directly connected to each other.

SPARQL allows for the creation of more complex queries from multiple repositories. It also allows for the creation of new graphs from the result. A full explanation of SPARQL falls outside the scope of this paper. See the SPARQL specification[8] for more detail.

So far we have described how to create descriptions of (local) networks and how to gather information from these descriptions. RDF also has a way to point to other documents, an example of this is shown in figure 6.

```
1  <ndl:Interface rdf:about="#Speculaas:port27">
2      <ndl:name>Speculaas:port27</ndl:name>
3      <ndl:connectedTo rdf:resource="nl:C6509:port7"/>
4  </ndl:Interface>
5
6  <ndl:Interface rdf:about="nl:C6509:port7">
7      <rdfs:seeAlso rdf:resource="http://www.netherlight.nl/ndl.rdf"/>
8  </ndl:Interface>
```

Fig. 6. Example of distributed repositories.

As shown before, lines 1 to 4 show the description of an interface of the `Speculaas`. However, note that in line 3, this port is defined to be connected to `nl:C6509:port7`. The `nl:` prefix defines that this interface is in the (XML) namespace[3] of *NetherLight*.

Lines 5 to 7 contain the definition of the interface `nl:C6509:port7`. The `rdfs:seeAlso` statement is used to link to the network description of Netherlight. An application or a crawler can follow this pointer to the description of NetherLight and get more information about this interface there.

4 Future Research

The next step in this research is to refine the Network Description Language and to develop tools to aid the development of these descriptions. Early results show that the creation of an initial network description requires a lot of effort, even though most information can automatically be extracted from networking equipment. However, once the description is created it can easily be maintained, and can help with network diagnostics and maintenance issues.

Security is an important issue with the description of networks. Not everyone wants to share their full network topology information. We will look into abstractions of the descriptions that provide enough information to be useful, but still hide the sensitive information from the outside. Through the use of automatic rewriting and AAA it may also be possible to filter the information based on the authorisation of the requester.

[3] The syntax as given here is a simplification: in an attribute value the URI of the namespace should be written out completely.

Another line of future research is to extend the Network Description Language to higher network layers. The end goal is to be able to describe the physical network, the state of the network and all services provided on the network.

5 Conclusion

In this paper we have introduced a way of applying the Resource Description Framework to describing networks by way of the Network Description Language. By using the linking capabilities of RDF, a distributed Topology Knowledge Base as described in [4] can be created.

The NDL and TKB unlock the potential of machine-readable metadata about the network for control-planes, service planes and other applications that require data about the network. The goals of NDL and TKB are to raise the degree of confidence in complex (hybrid) networks and to lower their operating costs.

The investment in codifying models and XML vocabularies pays dividend when new tools—e.g., bots and browsers—are set free to roam the synapses of the hybrid network across administrative boundaries. By using Semantic Web techniques, the tools themselves do not appear to be a significant R&D investment. Rather, early evidence shows that they can effectively be built upon commercial off-the-shelf and open source platforms.

6 Acknowledgements

The authors would like to thank TNO and the Gigaport project for funding this research.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Architecture for Differentiated Service, RFC 2475 (Informational) (Dec. 1998).
URL `http://www.ietf.org/rfc/rfc2475.txt`

[2] SURFnet, SURFnet6 lightpaths mark start of new Internet era (press release).
URL `http://www.surfnet.nl/info/en/artikel_content.jsp?objectnumber=107197`

[3] Resource Description Framework (RDF).
URL http://www.w3.org/RDF/

[4] F. Travostino, Using the semantic web to automate the operation of a hybrid internetwork, in: GridNets conference proceedings, 2005.

[5] User-Controlled LightPaths (UCLP).
URL http://www.canarie.ca/canet4/uclp/

[6] Dynamic Resource Allocation Controller (DRAC).
URL http://www.nortel.com/drac/

[7] The Semantic Web.
URL http://www.w3.org/2001/sw/

[8] E. Prud'hommeaux, A. Seaborne, SPARQL Query Language for RDF (2005).
URL http://www.w3.org/TR/rdf-sparql-query/

[9] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, RFC 3986 (Standard) (Jan. 2005).
URL http://www.ietf.org/rfc/rfc3986.txt

[10] Dublin Core Metadata Initiative.
URL http://www.dublincore.org/

[11] D. Beckett, B. McBride, RDF/XML Syntax Specification (Feb. 2004).
URL http://www.w3.org/TR/rdf-syntax-grammar/

[12] T. Berners-Lee, Notation 3 - an RDF language for the Semantic Web (1998).
URL http://www.w3.org/DesignIssues/Notation3

[13] Friend of a Friend (FOAF) Project.
URL http://www.foaf-project.org/

[14] FOAF-a-Matic.
URL http://www.ldodds.com/foaf/foaf-a-matic.html

[15] FOAFnaut.
URL http://www.foafnaut.org/