Knowledge-Driven Network Management - part 1

Rob Brennan
Networks are Growing in Complexity…

- Satellite
- Broadcast Networks (DAB, DVB-T)
- Core Network
- CDMA, GSM, GPRS
- 4G
- IP-based micro-mobility
- WiBro, HSDPA
- Bluetooth, Zigbee
- DMB
- Camera
- M-banking
- Navigation
- MP3
- Cam

- ISP
- Corporation
- ASP
- Access operator

Network Management and Ontologies © Rob Brennan
Growing Complexity…

Very large scales
  • million of entities

Ad hoc (amorphous) structures/behaviors
  • p2p/hierarchical architecture

Dynamic
  • entities join, leave, move, change behavior

Heterogeneous
  • capability, connectivity, reliability, guarantees, QoS

Unreliable
  • components, communication

Lack of common/complete knowledge
  • number, type, location, availability, connectivity, protocols, semantics, etc.
What are the key activities when engineering Network Management systems?

- Distributed Data Processing/Data Collection
- Data Integration
- Device/model/protocol heterogeneity
- Developing Data Analytics
- Developing NM Applications such as Network Control/Provisioning

… in the presence of network change
Properties of an Ontology or Knowledge-driven Approach

Flexible, self-describing, distributed “data structure”, a knowledge model

Have a formal semantics – ontology models are machine processable (vs UML models that are human-oriented)

Support

- Exchange of knowledge (via semantics) rather than data (via syntax)
- Data integration
- Schema evolution
- Distributed data publication
Semantics vs Syntax

**Semantics** = a way of encoding meaning (link between a term and a model of the world).

⇒ Good for building applications

**Syntax** = a way of encoding terms so that they can be distinguished, structured, grouped and related to each other in a grammar.

⇒ Good for building parsers

Note! We need a syntax (or syntaxes) for expressing a machine-readable semantics.
(Interlude) Interoperability Needs...

Semantics
Syntax is not enough!
Ontologies capture semantics in a machine-readable format, within a coherent logical model
Cat = Dog is not possible in a well-formed ontology

If we work with semantics, closer to human or business-level rules (policies)
    ie “make gold users high priority traffic”
This meaning can be encoded many ways
Ontology Definition

“an explicit and formal specification of a shared conceptualization”: (Gruber 1993)

• Explicit because it defines a set of concepts, properties, relationships, functions, axioms and restrictions.
• Formal = can be read and interpreted by machines.
  ie Enables “reasoning” = classification, inference
• It is a conceptualization because it is an abstract model and a simplified view of the entities it represents.
• Finally, it is shared because both agents communicating must use it

Ie Similar to an object-oriented class hierarchy

For more see: Natalya F. Noy and Deborah L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology
Properties of an (OWL) Ontology

Described by the family of languages in the W3C OWL (Web Ontology Language) Recommendation v2 2012
Built on top of W3C RDF/RDFS – to constrain it
Can build application-independent Reasoners that are generic programs for inference, consistency checking and querying of ontologies.
Can be queried using SPARQL
Graphs as knowledge – 1

How do we use graphs to represent knowledge?

A 'key' from which to hang the different facts

Don’t Panic

21:20::210509

fault1004

stackOverFlow

eventDetail

SoftwareError

2 hours

ProbableCause

EstimetoRepair

dateStamp

priority

maximum

21:20::210509

Cisco

router

serviceType

manufacturer

fromNetworkElement

Lucan Exchange

contains

contains

contains

basedIn

speciality

repairTeam3

ne03

ne02

ne01

contains

contains

contains

manufacturer

Lucan Exchange

蕲jar

Lucan Exchange

蕲jar

Lucan Exchange

蕲jar
Graphs as knowledge – 2

Things to note

• Scaling – the same graph can represent a lot of different knowledge simultaneously
• Agreement – you need to know what the various predicates 'mean'
• Structure – you need to know what nodes are related by a predicate
• Plurality – the same relationship may appear several times
• Symmetry – the same predicates can be used for common information, despite minor changes
• Asymmetry – relationships are inherently directed, which sometimes makes things awkward

...and this can get very tricky
...and this can be difficult to keep straight
For example both NetworkElements and Faults might have estimatedTimeToRepair
So a knowledge graph is inherently directed
Two ways to represent a graph

As nodes and arcs

- Nodes store facts, arcs store relationships between them

As triples

- A three-place relationship of 'subject, predicate, object'
- The node and edge structure is induced by the triples – each triple defines an edge, the different subjects/objects are the population of nodes, one node per individual string

Subject:  Predicate:  Object:
Fault1004  probableCause  SoftwareError
Graph based approach

The promise

• natural distribution
• easy merging
• naturally extensible
• easy publication and consumption
• easy querying (?)
• no need for null cells to represent absence
Semantic Web “Layer Cake”

Original

Data

RDF + rdfschema

XML + NS + xsd

Unicode

URI

Logic

 Ontology vocabulary

Rules

Proof

Trust

Digital Signature

Current [1]

User Interface/Applications

Trust?

Proof?

Unifying Logic?

Query:

SPARQL

Ontology:

OWL

Vocab:

RDFS

Rules:

RIF, SWRL

RDF

Turtle

Ntriples

N3

XML+NS+Schema

Unicode+URI

[1] adapted from John Sowa, Integrating Semantic Systems, 2010
Resource Description Framework (RDF)

RDF is a graph-based data model that allows us to identify things, classes of things and labelled parts of things based on URIs.

- “framework” => it does not make any assumptions about the types of things that it describes.
- It is the W3C’s meta-data standard
- It is NOT an XML standard, it does have a standard XML syntax for representing it in files/on the web.
- RDF not a knowledge standard *per se*:
  - A way of *defining* knowledge standards that maximises re-use on web
  - A way of defining knowledge graphs
  - No standard predicates
  - Tool support – editors, parsers, displays, …
What does RDF give us?

RDF is intended to address some of the issues we’ve identified in representing knowledge

- Extensible – easy to add new information at many locations, easy to define new predicates, entities
- Simple – minimises complexity by deferring it to RDFS, OWL
- Standard – defined by a standards body
- Scalable – used on the widest (Internet) scale
- Identity – URIs for subjects, predicates, objects allows us to independently define them uniquely at an internet scale

The way it accomplishes this is what we’ll look at next
RDF’s lazy approach to knowledge

“[In RDF] We know what we know, and we can ask for what we don’t know” – Alexander O’Connor

Open world assumption: if something isn’t stated, we don’t know if it’s true or not. (Compare to the closed world of DB.)

- But a lot of RDF does not use this!

No unique name assumption: things with different URIs could still be the same thing.
Basic structure – triples

RDF represents knowledge using a triple structure

- Subject
- Predicate
- Object

Knowledge is built up as a collection of these triples, contained within a (XML) file(s) or a “triple store” (RDF “database”) or published on the web.

- Many sources can make statements (with triples) about the same resources/entities.
- Applications can combine these triples into graph-based knowledge models (to answer questions about the world/the application domain).

Remember, a triple structure is one way of viewing a graph, so RDF essentially defines a knowledge graph.
Identifying “things” (resources)

RDF re-uses the URI as a global namespace of identifiers for things.
- Unique across entire WWW
- URIs can contain URLs that can be de-referenced (resolved) to find out more info about the “thing”
- If two resources use the same URI => they are the same thing

Back to triples:
- An RDF subject is always a resource => always a URI
- An RDF object can be a resource or a literal value
- What about predicates?
Recap: RDF URIs + Triples

Triple = subject  predicate  object
  e.g.  fault1    hasPrority    high

URIs are used to identify things, including predicates, e.g
  http://example.org/fault1  http://example.org/hasPrority  http://example.org/high

We can build graphs representing knowledge/data from sets of triples

http://example.org/fault1  http://example.org/hasPrority  http://example.org/high

Triples can be aggregated from all over the web… but?
Namespaces and URIs

Namespaces use URIs, and URIs can be made unique

- If I want to define a new structure I just define a namespace and assign it a unique URI
- If I own a domain I can give it any name I want under my domain name, secure in the knowledge that no-one else will (should!) use it

So a set of predicates, subjects, objects defined using a namespace can be uniquely differentiated from any other set of predicates across the entire web

- Cheap, decentralised model

Common sets of predicates may be given well-known names and URIs
Some common RDF namespaces

RDF:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
RDFS:  <http://www.w3.org/2000/01/rdf-schema#>
OWL:  <http://www.w3.org/2002/07/owl#>
XML schema  <http://www.w3.org/2001/XMLSchema#>
FOAF:  <http://xmlns.com/foaf/0.1/>

Two types of namespaces: slash(/) and hash(#)

- Hash implies a single file
  - Use for small, relatively static vocabularies

- Slash implies a set of files or dynamic generation (RESTful)
  - Use for large, dynamic vocabularies
Turtle – Terse RDF Triple Language

More human friendly/readable syntax
- Not XML based (just text)
- Does not have to represent a graph as a tree!

The same fact as before:

```turtle
@prefix ff: <http://www.fame.ie/ontologies/fame-faults#>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ff:fault1004 ff:additionalInfo "OK panic now" .
```

- Namespace declarations
- Subject, predicate, object separated by a whitespace all on one line
- Triple ends with a period
Turtle shortcuts

Multiple statements about the same subject:

```turtle
@prefix ff: <http://www.fame.ie/ontologies/fame-faults#>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ff:fault1004 ff:additionalInfo "OK panic now" ;
  ff:additionalInfo "This is really bad" ;
  ff:priority ff:high .
```

Multiple statements with the same subject and predicate:

```turtle
@prefix ff: <http://www.fame.ie/ontologies/fame-faults#>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ff:fault1004 ff:additionalInfo "OK panic now", "This is really bad".
```
A) Declare Prefixes/Namespaces

There is a standard namespace called fame faults (ff) used to describe all the entities in this system. It has a URI of http://www.fame.ie/ontologies/fame-faults#

ie line 1 of the turtle file is
@prefix ff: <http://www.fame.ie/ontologies/fame-faults#>

The standard RDF namespace URI is
http://www.w3.org/1999/02/22-rdf-syntax-ns#

ie line 2 of the turtle file is
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Now let’s write some Turtle! - 2

B) Describing instances

Use these predicates:

```
f:hasFault, f:hasId, f:hasSeverity, 
f:hasProbableCause, f:isConnectedTo
```

To write a set of triples to describe this system:

Network element 001 has a processing fault with an ID of 666, a timestamp of 010101 and a severity of high.

Network element 002 has a communications fault with an ID of 667, a timestamp of 010102 and a severity of medium.

Fault ID 667 was probably caused by fault ID 666.

Network element 001 is connected to network element 002.
RDF and RDFS

RDFS extends RDF with 'schema vocabulary', e.g.:

- Class, Property
- type, subClassOf, subPropertyOf
- range, domain (restricting types predicate attached to)

These terms add some structure/knowledge to the graph and how it can be interpreted:

```
ff:SoftwareError rdf:type rdfs:Class;
  rdfs:subClassOf ff:Cause.
ff:dateStamp rdf:type rdf:Property.
ff:additionalInfo rdf:type xsd:String.

ff:fromNetworkElement rdf:type rdf:Property;
  rdfs:domain ff:Fault;
  rdfs:range ff:NetworkElement.
```
OWL – Going Further

RDFS: Class, Instance, Property, Class hierarchy, Property hierarchy, Domains/Ranges, Annotations

OWL: ontology versioning, imports, distinguishes between data and object properties, Class definition in terms of other classes (union, intersection, compliment) and individuals (enumerations), cardinality restrictions, class disjointness, equivalence, union, Property types (inverse, transitive, reflexive, functional, symmetric), datatypes, instance keys, assertions (sameAs, differentFrom)
OWL Language(s)

Three species of OWL version 1

- **OWL full** is union of OWL syntax and RDF
- **OWL DL** restricted to the description logic FOL fragment
- **OWL Lite** is 'easier to implement' subset of OWL DL

OWL version 2 profiles (all subsets of OWL DL)

- **OWL EL** for efficient processing of large taxonomies
- **OWL QL** for efficient queries eg database integration
- **OWL RL** for rule-based reasoner implementation

OWL DL Benefits from many years of DL research

- Well defined **semantics**
- Formal properties well understood (complexity, decidability)
- Known reasoning algorithms
- Implemented systems (highly optimised)
Knowledge-Driven Network Management - part 2

Rob Brennan
Questions

What is the potential role of Ontologies in the Engineering of Network Management Systems?
What are the benefits and costs of engineering with ontologies?
Benefits of Engineering with Ontologies

Integration of key expert knowledge

Progressive modelling of elements
  • Progressive conceptual understanding

Easier integration of separately sourced models
  • Good for multi-vendor systems and value chains
  • Integrating with models from completely different domains, e.g. GIS, genetics etc.

Access to Linked Data Cloud

Logical consistency checks

Exploit general purpose reasoners and tools (Jena, Protégé etc)
Costs

Classic Bootstrap problem

- SemWeb: Someone has to bear the costs of providing semantic markup of content

Who is going to convert the models?
Who is going to integrate/map the models?

Ontological form may not be best for

- Transmission or storage
- Runtime-matching or interoperability
- Non-DL or rule-based reasoning, e.g. temporal reasoning, sub-reasoning

Mapping function May not be fully automated
Cost-Benefit Analysis

Benefits win out where:

- Grounding costs are minimised – especially where ontology used for normative/canonical form
- Service (re)composition and/or changes to policies and context are frequent and preferably dynamic

Costs win out where:

- Grounding cost are high: non-ontological languages used for core engineering modelling
- Frequency of change is low and resource/behaviour complexity is high
- Intra-organisational integration more common than inter-organisation
Recall TMF eTOM/NGOSS Vision

### eTOM maps the NGOSS Business View

<table>
<thead>
<tr>
<th>Strategy, Infrastructure &amp; Product</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy &amp; Commit</td>
<td>Fulfillment</td>
</tr>
<tr>
<td>Infrastructure Lifecycle Management</td>
<td>Assurance</td>
</tr>
<tr>
<td>Product Lifecycle Management</td>
<td>Billing</td>
</tr>
<tr>
<td>Marketing &amp; Offer Management</td>
<td></td>
</tr>
<tr>
<td>Service Development &amp; Management</td>
<td></td>
</tr>
<tr>
<td>Resource Development &amp; Management</td>
<td></td>
</tr>
<tr>
<td>(Application, Computing and Network)</td>
<td></td>
</tr>
<tr>
<td>Supply Chain Development &amp; Management</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enterprise Management</th>
<th>Business Process Analysis &amp; Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic &amp; Enterprise Planning</td>
<td>Business Process Map (eTOM)</td>
</tr>
<tr>
<td>Enterprise Risk Management</td>
<td>Conformance Testing &amp; Deployment</td>
</tr>
<tr>
<td>Enterprise Effectiveness Management</td>
<td>Compliance Testing</td>
</tr>
<tr>
<td>Knowledge &amp; Research Management</td>
<td>Solution Design &amp; Integration</td>
</tr>
</tbody>
</table>

- **Customer Relationship Management**
- **Service Management & Operations**
- **Resource Management & Operations**
- **Supplier/Partner Relationship Management**
- **Product Lifecycle Management**
- **Infrastructure Lifecycle Management**
- **Operations Support & Readiness**
- **Customer Relationship Management**
- **Service Management & Operations**
- **Resource Management & Operations**
- **Supplier/Partner Relationship Management**

**Key Components**
- **Knowledge Base**
- **Conformance Testing & Deployment**
- **Compliance Testing**
- **Technology Neutral Architecture**
- **Solution Design & Integration**
- **Shared Data Model (SID)**
- **Business Process Map (eTOM)**
- **Compliance Testing**
Where have ontologies actually been applied?

1. Network Management is about Network Resource Models (NRM/MIBs/…)
   - Ontologies are a type of flexible models

2. Network Management has too many NRM models!
   Ontologies are good for information/model integration
   eg integrating SMI, GDMO, CIM, 3GPP NRM models

3. Networks are heterogeneous
   Models (above)
   Staff knowledge

Where have they been applied? Part 2

Policies are rules, another form of logic

Semantic Web logical framework for data/classification (structured knowledge), it also has support for unstructured knowledge in the form of rules eg SWRL, RULE-ML, …

⇒ Can encode policies in a single framework, a rule-based reasoner

In general unstructured rule-bases do not scale well

 Addition of structured knowledge helps with scalability/predictability/policy analysis/policy integration
Areas where Ontologies have been applied

Fig. 1 Ontology applications to network and system management [9]
“Semantic Management”

It has an inference engine to check constraints.

Each provider or gateway loads its mapping ontology (M.O.) for its information model.

Manager

Common model


M.O. CORBA gateway

M.O. CMIP gateway

M.O. SNMP gateway

M.O. DMI gateway

IIOP CMIP SNMP IPC/RPC

CORBA agents CMIP agents SNMP agents DMI agents

Fig. 2 Proposed architecture for the semantic management [9]
Policy Interoperability
Behavior Definition in Web Service Management Interfaces

Fig. 4 Application of service ontologies to network management
Example Service Description

@prefix frm: <http://fame.ie/federalrelationshipmanager> .
@prefix dc: <http://purl.org/dc/elements/1.1>/ .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix ex: <http://example.org#> .
ex:MyBigTV frm:hasAuthority ex:MySharedUpnpServices ;
    rdf:type frm:UpnpOverXmppService ;
dc:creator ex:TheHanOwner ;
dc:date 2010-01-14;
rdfs:comment "Capability description for HD TV in front room" ;
frm:generatesEvent frm:ConfigurationError ;
frm:generatesEvent http://sw.opencyc.org/concept/Mx4rwJN-YpwprcN5Y29ycA ;
frm:serviceName "Display on my big TV" ;
frm:serviceType dbpedia:Video_Mixing_Renderer ;
frm:hasInput dbpedia:Streaming_media .
Federations – A general purpose layered approach

- **Trusted Communication**: Ensure that parties establish and maintain trusted communications (as per their security requirements).
- **Relationship Definition**: Defines the rules for cooperation. What are the decision-making and conflict resolution rules and the conditions of membership.
- **Shared Semantics**: Establish and maintain shared semantics and models of resources, policies, services, etc.
- **Operational Rules**: Defines which services and information resources (and constraints) are shared by parties and are available to other parties to the federation.
- **Shared Resources**: Defines the configuration of shared resources at any particular time during operations.
- **Monitoring & Auditing**: Analyses operations over time and evaluates compliance with policies defined at lower levels.
TCD FRM Prototype

Local Domain

Capability Delegation Arc

Federation

Capability Sharing Arc

FRM Server

External Domain

- Defines Monitoring Ontology for Federated Infrastructures (MOFI)
An Ontology-Based Information Extraction System for bridging the configuration gap in hybrid SDN environments, A. Martínez et al. 2015

- OpenFlow depends on NETCONF which lacks management models => extract them from CLI tools
Lessons Learned 1

OWL as an Ontology Language
- Lack of proper integration with rule
  + supports distributed ontologies/instance data
  + supports richer queries (via SPARQL) than SQL
  + SPARQL give remote query support

Ontology Reasoning
  + general purpose reasoners work on many domains
  + classification features not explored enough
  + theoretically could be used to capture network control knowledge
  + useful for policy analysis
Lessons Learned 2

SWRL as a Rule Language
- no NOT or OR operators
- only monotomic inference
bad for handling anonymous classes

Semantic Mapping (Interoperability)
+ often used

Web Service Definitions
+ theoretically useful
- OWL-S never standardised
Conclusions

Networking application developers will benefit from the information model interoperability, higher design expressiveness, and the availability of semantic web execution environments. System integrators will find it easier to compose their management solutions due to the extensibility and interoperability of ontology models, as well as from the real-world view that allows them to design at a higher abstraction level.

Network administrators will not only benefit from more powerful applications, but the real-world view and ease of use of this framework will allow them to participate more directly in the higher level application design, being able to quickly transfer their expert knowledge into the management applications, and in this way automating more and more network management tasks.

This idea is also reinforced due to the availability of more general purpose ontology-related tools than network management ones.
But…

(a) if ontologies are used as an umbrella model that try to integrate heterogeneous definitions, the complete semantic harmonization of the different information models in an ontology cannot be made fully automated (a human expert is needed to do the final validation of the semantics mappings)

(b) the inclusion of formal management behavior definitions with ontology-related behavior languages like SWRL is difficult due to very basic expressiveness of dynamic behavior that these languages have.
Recap: How can ontologies be applied to network management?

Strengths and weaknesses of semantic modelling
Challenges in network management suitable for the application of ontologies
Examples of recent research or trial systems for ontology-based network management
Relationship of knowledge modelling to the NGOSS architecture standardized at the TM Forum.