In: Semantic Web: Standards, Tools and Ontologies ISBN 978-1-61668-471-6
Editor: Kimberly A. Haffner

Chapter 3

KNOWLEDGE MAPS FOR CAMPUS IP NETWORKS: FROM RELATIONAL DATABASES TO RELATIONSHIP-CENTRIC SEMANTIC MODELS

L. I. Lumb*[*]*, R. Gorsht & D. Zeng,
University Information Technology, York University
4700 Keele Street, Toronto, Ontario M3J 1P3, Canada

Abstract

Campus IP (Internet Protocol) networks have various management requirements. To address a number of these requirements, York University makes use of Open Source Netdisco. In addition to discovering the IP network requiring management, Netdisco extracts management information via Simple Network Management Protocol (SNMP), and serves as an aggregation repository for the same. The combination of a discovered network and device-specific information allows Netdisco to produce topology maps. Unfortunately, however, Netdisco's capabilities are limited in this important area. Rather than extend Netdisco directly, the approach taken here is one of applying emerging standards and implementations from the Semantic Web. Using Netdisco as its data source,

[*]E-mail address: ian@yorku.ca

a relationship-centric representation in the Resource Description Format (RDF) is automatically derived via an enabling implementation based on the Redland RDF Library. As a consequence of this new representation, the RDF-based model can be queried using SPARQL to produce knowledge maps of the Campus IP network; since these knowledge maps are not restricted to conveying topological relationships, they are regarded as an important generalization afforded by use of Semantic Web standards and implementations. When viewed in the broader context of a Semantic Framework, additional possibilities arise — e.g., the introduction of editorial metadata via annotations in XML Pointer Language (XPointer). This framework also provides the means for automating the extraction of a vocabulary for IP network knowledge cast as an RDF Schema, thus extending the utility of previous efforts to represent SNMP management information in the eXtensible Markup Language (XML). Finally, in the even broader context of the Semantic Web, the approach taken allows for incremental-to-substantial progress against some of the key obstacles currently impeding realization of the promise of the Semantic Web.

1. Introduction

Arguably, "promise versus reality" is the sub-text that has characterized the Semantic Web since its inception. And although significant progress continues to be made in closing this gap, a recent compilation by Google researchers [17] identifies the following four areas in which significant hurdles remain:

Ontology writing Regarded as an area in which significant progress has been made, there remains [17], however, "... a long tail of rarely used concepts that are too expensive to formalize with current technology."

Difficulty of implementation In moving beyond static Web pages, the non-specialist's implementation challenge is stated via example [17]: "Creating a database-backed Web service is substantially harder, requiring specialized skills. Making that service compliant with Semantic Web protocols is harder still."

Competition Of standards, it is often bemoaned [54]: "The wonderful thing about standards is that there are so many of them to choose from."[1] In-

[1] Often cited, there remains some dispute as to the origin of this quotation - see, e.g., [54].

creasingly, this is the case with ontologies. And as if this is not a significant enough hurdle in-and-of-itself, there is concern of a less-than-altruistic element possible within the context of competition as a hurdle.

Inaccuracy and deception Trust is placed at the pinnacle of Berners-Lee's 'stack of expressive power' [5, Slide 27]. Given that the Web is an environment in which "Anyone can say Anything about Any topic",[2] despite this increasingly solid technology foundation, trust is not easily established owing to unintentional-to-malicious inaccuracies or deceptions.

In the spirit of addressing the Semantic Web's promise-reality gap, attempts are made here to target aspects of the hurdles identified above via a use case from the management of Internet Protocol (IP) networks. Thus the following section (§2.) establishes the motivation for the current investigation via the use case; in part, a relational database is identified as a repository of data that needs to be exposed to the standards and implementations of the Semantic Web. Through the establishment of an automated means for extracting data from this network-management database, and reformulating it in a relationship-centric representation (§3.), progress is made against the implementation-difficulty hurdle identified above. In crafting this relationship-centric representation in the Resource Description Format (RDF, [1, 39, 68]), a more-expressive basis for modeling and manipulating (e.g., via queries in SPARQL, [69]) IP network data is established. Key to successfully addressing implementation difficulties is use of the Redland RDF Library [12]. The process of extracting data from a relational database and representing it in RDF is cast within a broader, previously established Semantic Framework in §4. Because the outcome of this systematic enhancement in expressivity is an informal ontology, aspects of the ontology-writing hurdle are addressed. Handled via annotation, editorial metadata also falls within the scope of this framework. And when this editorialization is afforded a social context, there exists the potential for progress against the hurdles of competition and trust. An additional application of the Semantic Framework affords a generalization of the instance-oriented efforts of previous sections in §5. In the development of a vocabulary for IP network knowledge, a procedure for extracting schema definitions from pre-existing network management documents (i.e., Simple Network Management Protocol (SNMP, [56]) Management

[2] This statement is known as the Web's *AAA Slogan* [1].

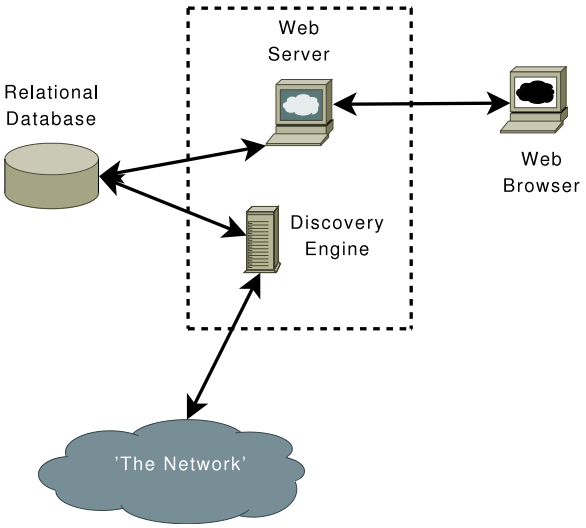

Figure 1. A simplified schematic for the architecture of Netdisco (after [4]).

Information Bases (MIBs, [38])) is shared. Again such procedures, when fully automated, have the potential to simultaneously address ontology-writing and implementation-difficulty hurdles while ensuring measures of trust — as SNMP MIBs, as standards, are already trustworthy. Work related to that presented here is reviewed briefly in §6., before conclusions are drawn in the final section (§7.).

2. Network Management via Netdisco

Netdisco is described as a tool for network management [13]. Based on the inventory of an IP network, Netdisco allows for the management of switch ports. In a security-conscious fashion, Netdisco facilitates actions on switch ports and provides an audit trail of the same. Schematically illustrated in Figure 1, Netdisco is a three-tier client/server application. The middle tier is comprised of two components. The first component is a Web server whose role is to facilitate client-driven interactions within and beyond the middle tier. Within the middle tier, the Web server interacts with the Discovery Engine; the Discovery Engine is described in detail below. Beyond the middle tier, the Web server facilitates

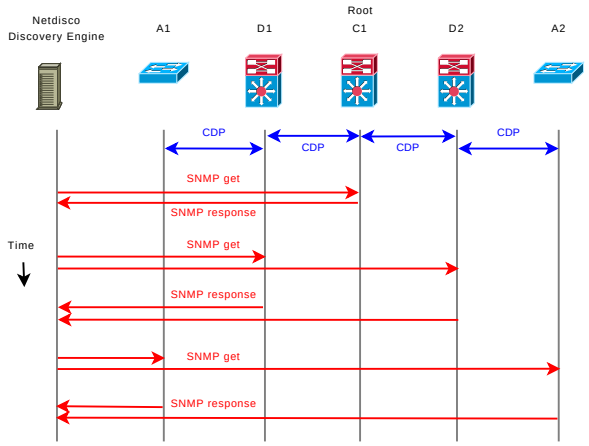

Figure 2. Real-time network discovery in Netdisco.

interactions with a relational database at its back end, and Web browsers at its client-facing front end. In terms of implementation, Netdisco makes use of a Mason-enhanced Apache Web server [48]. Given that Netdisco makes extensive use of Perl [51], the pairing with Mason is a very logical one, as Mason is a Perl-based Web site development and delivery engine [22]. At the back-end, Netdisco employs PostgreSQL [52] as its relational database. Because Netdisco is rendered to the client tier in HTML, it works with almost any Web browser.

The second component comprising the middle tier is the Discovery Engine. On a scheduled basis, the Discovery Engine transits the network via requests and responses using SNMP. To automate the discovery process, the targets of the SNMP requests are identified through the use of a bootstrapping approach. First, discovery is initiated by identifying to Netdisco the root of the IP network; in Figure 2, this is the core node[3] `C1`. Because each switch is aware of its neighbors via Cisco Discovery Protocol (CDP, [55]), SNMP interactions with the `C1` switch expose this information; this is how, for example, switches `D1` and `D2` are identified next in Figure 2. Because each switch can identify its neighbors in this fashion, a neighbor-by-neighbor inventory of all switches on

[3]Note that use is made here of the three-layer architecture for campus networks championed by Cisco Systems Inc. In this architecture, the campus network is comprised of core, distribution and access layers.

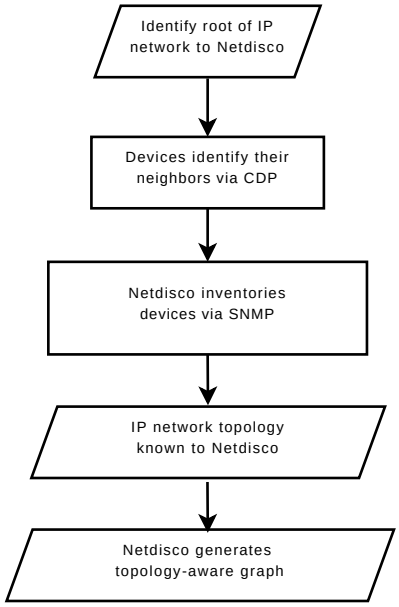

Figure 3. A flowchart summarizing the Netdisco process for generating a topology map of an IP network.

the network is generated in real time. Again, in reference to Figure 2, this is how access-layer switches `A1` and `A2` are also 'discovered'. Armed with this knowledge of the network, Netdisco is now able to make a sequence of targeted SNMP requests and receive the corresponding responses. More specifically, Netdisco employs the `SNMP::Info` Perl module [47] to interact with network devices and retrieve the data stored in the corresponding SNMP MIB.

By identifying the root of the network, and then successively identifying neighbors, Netdisco is able to map the topology of the network; this process is summarized by the flowchart presented in Figure 3. Thus network maps are a natural byproduct of the discovery process. In addition to illustrating the Netdisco client interface (i.e., the client-tier front end alluded to previously and specifically in Figure 1), Figure 4 provides an illustrative example of a Netdisco network map for one of York University's satellite locations. From the network map, it is clear that `nadal-1.swx` provides connectivity for four ac-

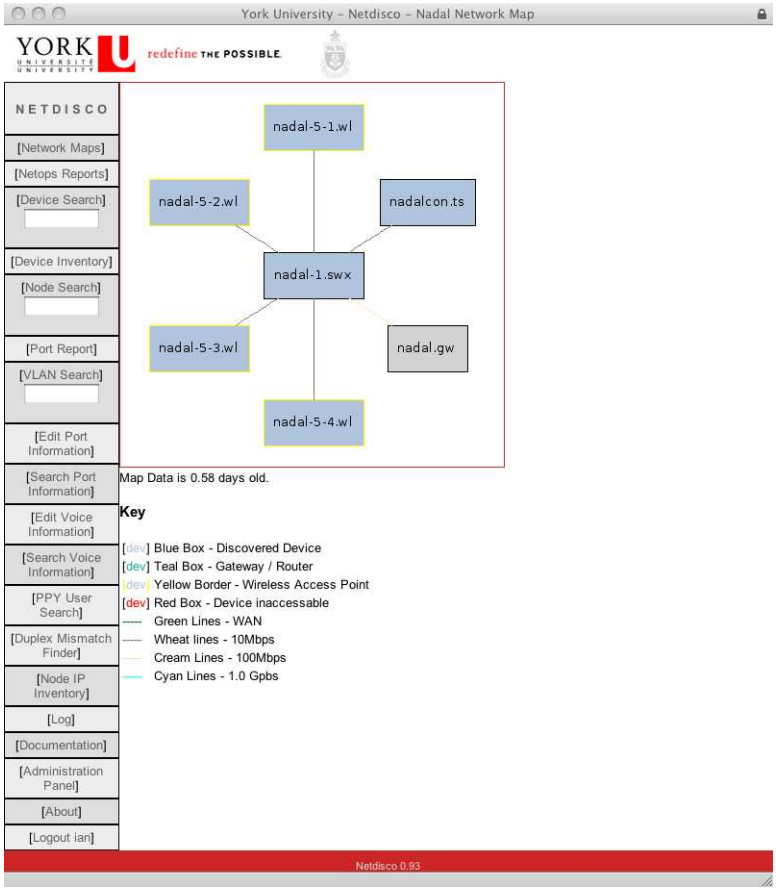

Figure 4. An IP network map of The Miles S. Nadal Downtown Management Centre at York University via Netdisco.

cess points[4] `nadal-5-*.wl` plus a terminal server (`nadalcon.ts`), and is uplinked to a gateway interface `nadal.gw`.[5] Network maps in Netdisco are

[4]Note that wireless access points are known to Netdisco via SNMP because they (the access points) also support CDP; this applies to both legacy wireless access points instantiated in an autonomous mode as well as those same access points enabled in a light-weight mode.

[5]Fully qualified domain names are produced by adding the suffix `.yorku.ca` to each of the device names identified here.

actually image maps. Thus clicking on `nadal-1.swx` in Figure 4 results in the details on this switch as revealed in Figure 5. The data presented in Figure 5 is retrieved from the Netdisco PostgreSQL database. And although the database caches the most-recently discovered data via the SNMP-based interactions described previously, sites can optionally include additional data which is also stored in the database.

Clearly Netdisco has the potential to be very useful in delineating IP network topology. Consider now, in contrast to the network map for this relatively small satellite location at York University (i.e., Figure 4), the illustration in Figure 6 for the main campus. As was the case in Figure 4, Figure 6 is also an image map when rendered via a Web browser. Even in case of the Web-browser rendering, however, Netdisco's attempt to represent the sheer magnitude of the environment is largely ineffective. In accurately representing the topology of the network at York's Keele Campus, Netdisco presents a perspective that largely confirms the implementation of a campus network architecture in terms of core, distribution and access layers — but not a lot more. It would be highly advantageous, however, to depict this network from different perspectives. For example, instead of revealing the topology of the entire IP network at the Keele Campus, it might be a requirement to illustrate just the core and distributions layers as in Figure 7. Unfortunately, this latter perspective (i.e., Figure 7) was manually generated through human intervention by a network specialist. Eminently clear to a network specialist (i.e., an IP-networking-knowledgeable human), such abstractions are beyond the scope of Netdisco (i.e., an otherwise unempowered machine).[6] The more-significant challenge, however, and the one focused on for the remainder of this contribution, is that of making such abstraction requests expressible in a machine-processible form. In other words, one objective of the current effort is to automatically generate topology maps such as Figure 7 through the expression of appropriate requirements to a machine.

[6] And although some deft scripting may be required to modify Netdisco to provide such depictions, there is very little question that such modifications are achievable ... from the graphing perspective.

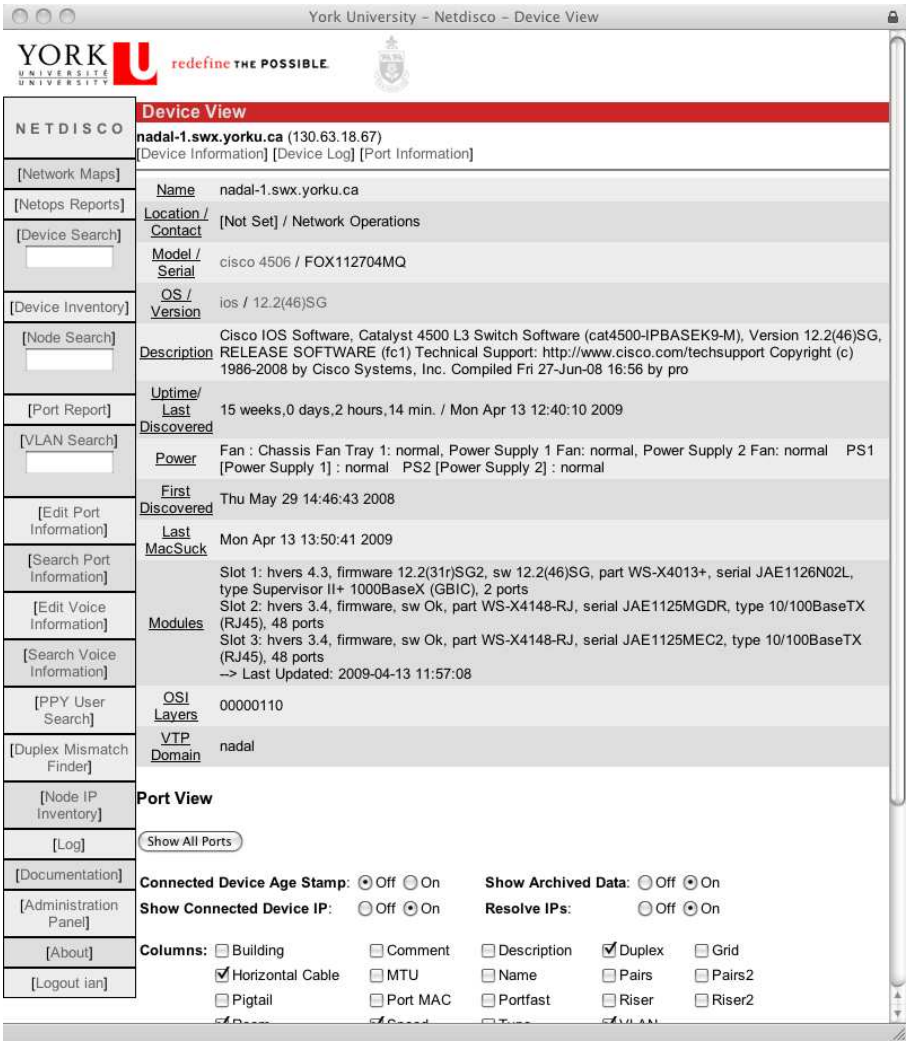

Figure 5. Netdisco specifics for the `nadal-1.swx` at the Nadal Management Centre at York University.

3. Relationship-Centric Representations

Written in Perl and made available under a liberal Open Source license, Netdisco offers significant opportunity for customization. That being the case, it

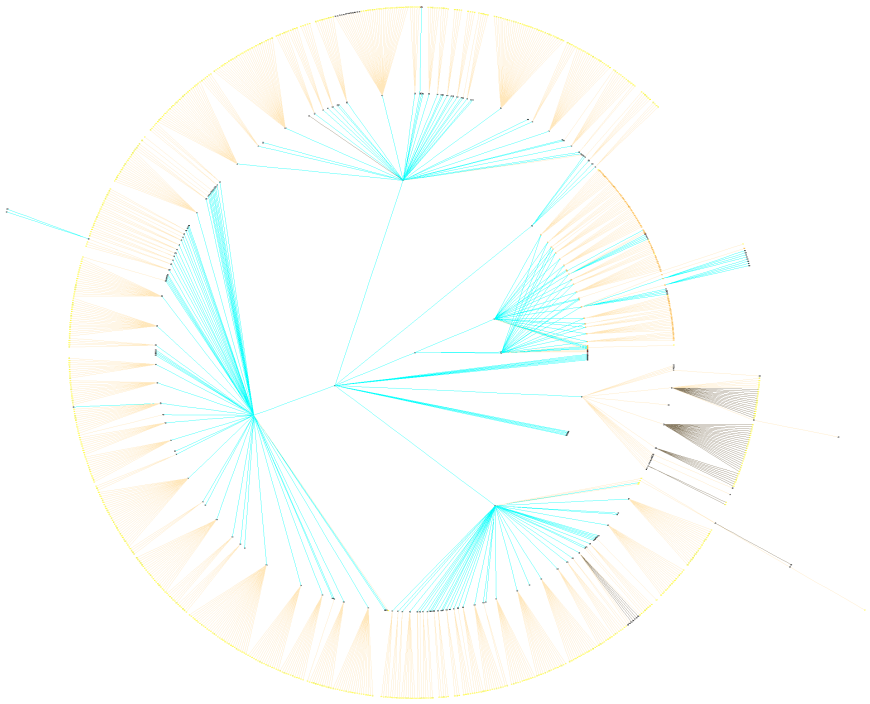

Figure 6. Netdisco topology map for the IP network at the Keele Campus of York University. Note that only the image is displayed; the Web browser and Netdisco user interface has been omitted.

is almost certain that Netdisco could be enhanced to deliver topological maps along the lines of Figure 7. Using this approach, the *semantics* of the topological view required by the network specialist would need to be encoded via Perl for use within the Netdisco context. The network specialist's requirement for even a slightly different topological view would necessitate modification at the source-code level. And as the need for different views increases, management of the corresponding source code emerges as a challenge.

This source-code-centric approach underlines an *expressibility gap* — i.e., that which is known to humans (the network specialist) is not known to machines (Netdisco). Thus the approach taken here is to bridge this expressibility gap through the introduction of a relationship-centric approach for represent-

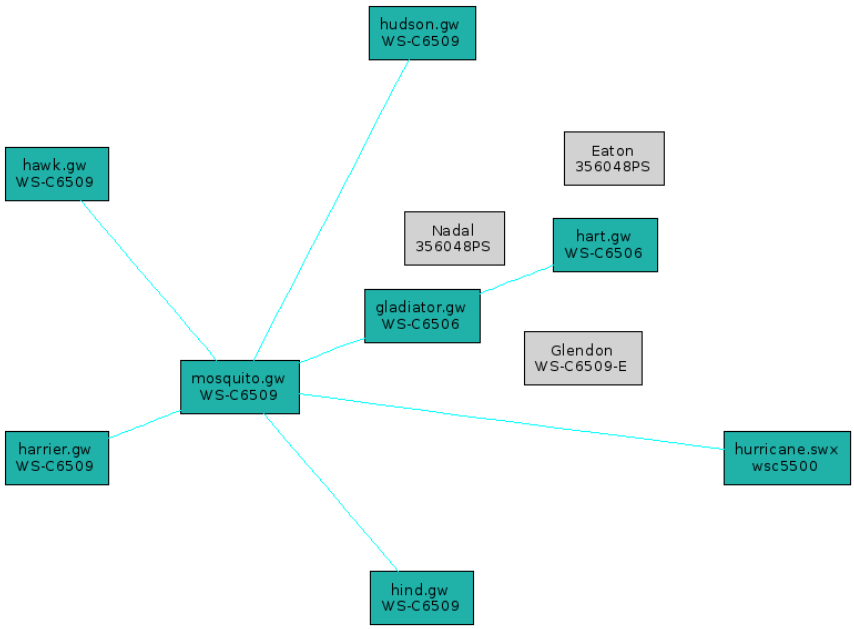

Figure 7. Netdisco topology map for the core and distribution layers of the IP network at the Keele Campus of York University.

ing knowledge. As shown in Figure 8, the approach employed here takes full advantage of the effort expended by Netdisco. In fact, Netdisco's relational database is regarded as the primary data source for the IP network — i.e., for network components, plus their associated descriptions and topologies. This data regarding the network is used to build a relationship-centric representation. Because relationships are explicit in this representation, the above requirement for custom code is replaced by queries articulated in an appropriate language. The results of these queries can then be expressed graphically. In the remainder of this section, the details of the relationship-centric approach are elucidated.

3.1. Introducing Relationship-Centric Representations via RDF

To introduce the relationship-centric representation, and establish the necessary context with the content provided in §2., suppose that Table 1 is a snippet of a

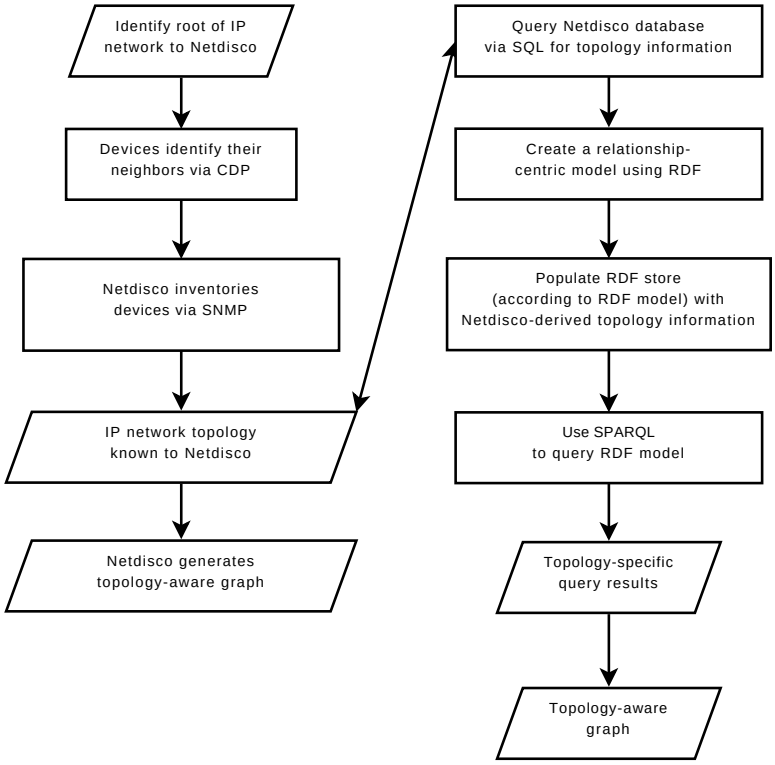

Figure 8. A flowchart summarizing the process for generating network-topology maps.

table in Netdisco's relational database.[7] In Table 1, each row corresponds to a network component being accounted for as a record in the Netdisco database.[8] In the same Table, the columns `IP Address`, `IP Name` and `IP Device`

[7] In proceeding, it is important to note that the current supposition *grossly* oversimplifies the schema in use by Netdisco, and is being used in the current context for the purpose of illustration only. In other words, this example does not capture a table actually present in Netdisco. Also, in the current context, the notion of a key field has been omitted.

[8] Netdisco identifies network components as devices (e.g., switches) or nodes (e.g., servers).

`Type` enumerate fields in the database for each of the records.

Table 1. An illustrative snippet from a table of Netdisco's relational database.

IP Address	IP Name	IP Device Type
130.63.29.11	hawk.gw	6509
130.63.29.12	mosquito.gw	6509

Consider now the specific value `mosquito.gw` in Table 1. In a relationship-centric representation, `mosquito.gw` is the *object* of the relationship whose *subject* is `130.63.29.12` and whose *predicate* is `IP Name`. In other words, relationships are captured as subject-predicate-object triples that can be serialized in various ways and illustrated graphically in the manner of Figure 9. Although Figure 9 encapsulates the same semantics as does Table 1, there are some significant differences that require explanation. Perhaps most obvious in Figure 9 is the introduction of Uniform Resource Identifiers (URIs, [11]). Specifically, `http://www.yorku.ca/cns/netops/nkm/` has been prepended to the IP addresses for the two devices whose relationship has been captured by the third URI. Through unique namespace declarations, URIs make explicit the devices in question (`http://www.yorku.ca/cns/netops/nkm/130.63.29.12` and `http://www.yorku.ca/cns/netops/nkm/130.63.29.11`) and their relationship (the former is the `http://www.yorku.ca/cns/netops/nkm/1.0/neighbor` of the latter). Furthermore, the notion of a `neighbor` is a unique concept expressed in version `1.0` of the `nkm` namespace.

A more-complete, graphical representation of the subject `http://www.yorku.ca/cns/netops/nkm/130.63.29.12` from Table 1 is presented in Figure 10. In addition to the expression of one switch as the `neighbor` of another, note the appearance of URIs asserting the Table 1 concepts IP Name and IP Device Type as the predicates `http://www.yorku.ca/cns/netops/nkm/1.0/ip_name` and `http://www.yorku.ca/cns/netops/nkm/1.0/ip_device_type`, respectively. The respective objects for each of these predicates are

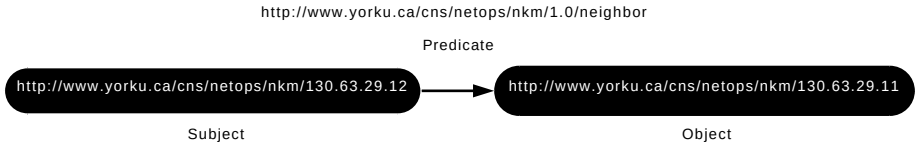

Figure 9. Graphical representation of a specific subject-predicate-object triple taken from Figure 1.

`mosquito.gw.yorku.ca` and `6509`.

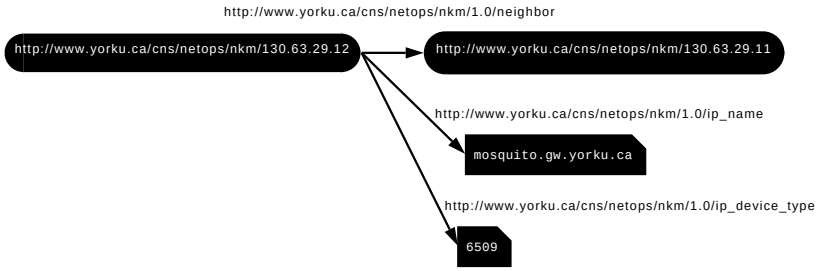

Figure 10. More-complete graphical representation of a specific subject-predicate-object triple taken from Table 1.

Listing 1 presents a serialization of the subject `http://www.yorku.ca/cns/netops/nkm/130.63.29.12` from Table 1 that was then represented graphically in Figures 9 and 10. From the very first line of this Listing, it is clear that the serialization is implemented in XML. More specifically, use is made of RDF, and the serialization in evidence is RDF/XML. In addition to the declaration of the World Wide Web Consortium's (W3C's) namespace for RDF (line `3` of the Listing), also evident are two namespaces created for current purposes (lines `4-5`). Note also in these declarations use of variables — e.g., `nkm` is used elsewhere in this Listing as a place holder for `http://www.yorku.ca/cns/netops/nkm/1.0/`. Such efficiencies are readily evident in the three `rdf:Description`'s that follow regarding the subject `http://www.yorku.ca/cns/netops/nkm/130.63.29.12`. Using the `rdf:about` element, the corresponding `IP Name`, `neighbor`

and `IP Device Type` are presented in context; in other words, humans[9] *or* machines that parse this serialization 'know' that there exists a relationship between the subject and its associated predicate-object pairings. From these examples it is clear that the variables `rdf` and `nkm`, and the namespaces for which they are place holders, are used to disambiguate the context of the elements appearing in RDF/XML serializations such as this one. If neither `rdf` nor `nkm` appears explicitly in the specification of a tag, use of the `xml:base=''http://www.yorku.ca/cns/netops/nkm/''` declaration ensures that `http://www.yorku.ca/cns/netops/nkm/` is the default namespace.

```
1 <?xml version="1.0" encoding="utf-8"?>

3 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns↘
  →#"
4          xmlns:nkm="http://www.yorku.ca/cns/netops/nkm/1.0/"
5          xml:base="http://www.yorku.ca/cns/netops/nkm/">

7 <rdf:Description rdf:about="130.63.29.12">
8    <nkm:ip_name>mosquitolb.gw.yorku.ca</nkm:ip_name>
9 </rdf:Description>

11 <rdf:Description rdf:about="130.63.29.12">
12    <nkm:neighbor rdf:resource="130.63.29.11"/>
13 </rdf:Description>

15 <rdf:Description rdf:about="130.63.29.12">
16    <nkm:ip_device_type>6509</nkm:ip_device_type>
17 </rdf:Description>

19 </rdf:RDF>
```

Listing 1: RDF/XML serialization of subject `130.63.29.12` from Table 1.

To independently confirm that the serialization in RDF/XML is syntactically correct and valid, Listing 1 was parsed by the W3C's RDF Validation Service [71].[10] Although the correctness of the RDF/XML of this Listing is not

[9] Note that extra whitespace has been introduced to improve the readability of the Listing for human consumers.

[10] Note that use of valid URIs is not a requirement of this service.

shared directly here, a graphical representation of the outcome is presented in Figure 11. On comparing and contrasting this Figure with Figure 10, it is clear that the desired level of consistency has been achieved. Stated differently, there is a reciprocity between the graphical and RDF/XML serializations of relationships in RDF. In passing, it is important to note that Figures 9, 10 and 11 are *not* topological representations of a fragment of the York IP network. Rather, these figures are a means for visualizing graphically relationships that exist in this domain of interest. And though it is certainly true that one of the relationships does represent a topological notion (i.e., `neighbor`), the other two represented relationships (i.e., `IP Name` and `IP Device Type`) have no innate value in conveying topological context. This not-so subtle distinction must be kept clear in the remainder of this chapter.

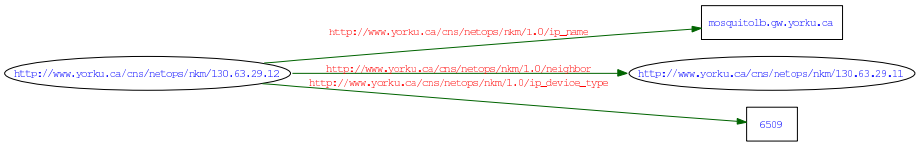

Figure 11. Graphical representation of the RDF/XML serialization presented in Listing 1 via the W3C RDF Validation Service [71].

It is possible to manually create the RDF/XML markup presented in Listing 1. Of course, this rapidly becomes an excessively tedious task as the number of relationships requiring representation increases to numbers anything close to being useful in the current context. Fortunately, as is demonstrated in the following section, it is possible to automate such tasks.

3.2. Implementing Relationship-Centric Representations via Redland

As noted above, it is possible to produce RDF/XML serializations such as that of Listing 1 by hand coding. Fortunately, that is not the only means for generating such results. Therefore, consideration is given here to the automated scheme depicted by the workflow in Figure 8.

To fix ideas, the approach used here regards the Netdisco database as the source of data for the relationship-centric representations that are to be devel-

oped. For a given root device, an implementation of Algorithm 1 in Perl extracts the `IP Address`, `IP Name` and `IP Device Type` for all identified neighbors from the Netdisco database. Interaction between the Perl script employed and the Netdisco database is facilitated through use of the `Perl::DBI` Perl module [53]. This Perl module abstracts away the low-level details of database interaction, and allows emphasis to focus on the indicated query to be captured and executed against Netdisco's PostgreSQL database. Other than use of SQL [57], Algorithm 1 is language neutral. In the remainder of this section, it will be shown that the same neutrality exists elsewhere in the implementation of the relationship-centric representation; this neutrality allows developers to exercise considerable choice in building the corresponding technology platform.

Algorithm 1 Pseudo-code skeleton of Netdisco-oriented algorithm for discovery of an IP network via neighbors.

```
{Invoke external packages - e.g., to facilitate interaction with databases}
{Accept IP address of root device from standard input}
{Define variables - including the neighbor's IP address, IP name and IP device type}
{Define database specifics - including server and port for access}
{Open connection with the database}
{Enter query in SQL to extract network neighbors from a given root device}
select ip, port, remote_ip, remote_port from device_port where ip='$ip'
   and remote_ip is not null
{Execute SQL query}
while (the SQL query returns results) do
   {Acquire additional details on the neighbor via additional SQL queries}
   {Print to standard output the neighbor's IP address, IP name and IP device type}
end while
{Close connection with database}
```

Algorithm 1 allows for the extraction of neighbors and their associated RDF triples from Netdisco's relational database. The next step identified by the workflow depicted in Figure 8 requires that this data be cast in a relationship-centric

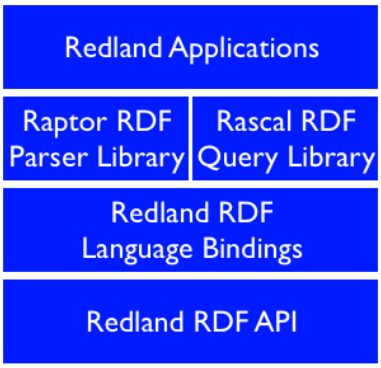

Figure 12. The Redland stack (after [12]).

representation based on RDF. Given that Netdisco is written in Perl, efforts were made to seek an RDF platform sharing the same affinity for Perl. Although it is not written in Perl, the Redland RDF Library [12] provides language bindings in Perl,[11] and certainly satisfies current needs for a Perl-based platform for RDF development.[12] In addition to the language bindings, Redland includes the Raptor RDF Parser Library and the Rascal RDF Query Library. Thus the language bindings (and underlying RDF API), in tandem with the parsing and querying libraries, collectively comprise a development stack (see Figure 12) that allows for the creation of 'Redland applications'.

The algorithm for one such Redland application is presented via a pseudo-code representation here as Algorithm 2. The implementation developed for current purposes makes use of the `RDF::Redland` Perl module as the key external package. Anticipating RDF/XML serializations along the lines of Listing 1, the required predicates are declared as URIs using a built-in Redland construct, namely `RDF::Redland::URI`. For a variety of reasons, Redland allows the RDF representation being created to be stored (via

[11] Written in C, Redland also provides bindings for C#, Java, PHP, Python, Ruby and Tcl [12]. This means, for example, the RDF representation created via Perl in the current context can be further manipulated via Redland's language bindings for Ruby. Such flexibility offers promise for a number of possibilities.

[12] Of course, alternatives to Redland are available. One popular alternative based on Java is the Jena framework [20].

Algorithm 2 Pseudo-code skeleton for the RDF-representation algorithm (modeled after a usage example in Perl provided with Redland [12].)

```
{Invoke external packages - e.g., RDF platform}
{Define variables - including the URIs for the required predicates}
{Instantiate a persistent database for this RDF representation}
{Instantiate an RDF model for this representation in the persistent database}
while (executing Algorithm 1 delivers data to standard output) do
   {Convert standard output into RDF subject-predicate-object triple}
   {Insert RDF triple as statement addition to RDF model}
end while
{Instantiate RDF/XML serializer}
{Establish namespaces for RDF/XML serialization}
{Serialize the RDF model to a file as output}
{Close all open files}
{Deconstruct the serializer}
{Deconstruct the persistent data store}
{Deconstruct the RDF model}
{Visualize results in the .rdf file graphically}
```

`RDF::Redland::Storage`) in a persistent database. Although Redland offer various possibilities, use was made of the Berkeley/Sleepycat DB. The data that has been extracted from the Netdisco database, and transformed into a relationship-centric representation, is cast into the Redland RDF store as an RDF model (via `RDF::Redland::Model`). More than simply a means to populating Redland's RDF store, Redland's RDF model is primed to be established as the embodiment of the relationship-centric representation that is sought after. With the Redland-based platform ready to accept data, execution of the `while` loop in Algorithm 2 continues for as long as execution of Algorithm 1 outputs data extracted by querying Netdisco's database. Each line of output from the Netdisco database query is systematically converted to an RDF subject-predicate-object triple and then added to the Redland-based RDF model. With all of the output processed, the `while` loop terminates, and execution focuses on serializing the output as RDF/XML. Once the Redland serializer has been instantiated in RDF/XML mode, and additional namespace declarations have

been established,[13] the serializer outputs the Redland RDF model to a `.rdf` file — along the lines of Listing 1. The remainder of Algorithm 2 focuses on closing open files, gracefully decommissioning the Redland environment, and visualizing the results. Although visualization of the results is included in Algorithm 2, at the moment this is a step handled outside of the Redland platform. Specifically, at this stage, results along the lines of Figure 11 are validated and visualized using the W3C RDF Validation Service [71].

A modified version of Figure 1 is presented in Figure 13. This latter schematic illustrates the addition of the Redland platform to the existing Netdisco environment. After installing the Berkeley/Sleepycat DB via a package-management utility, it is possible to build Redland from source or install a pre-compiled version also using a package-management utility. Collectively, this means that a Redland RDF platform can be established and put to use efficiently and effectively.

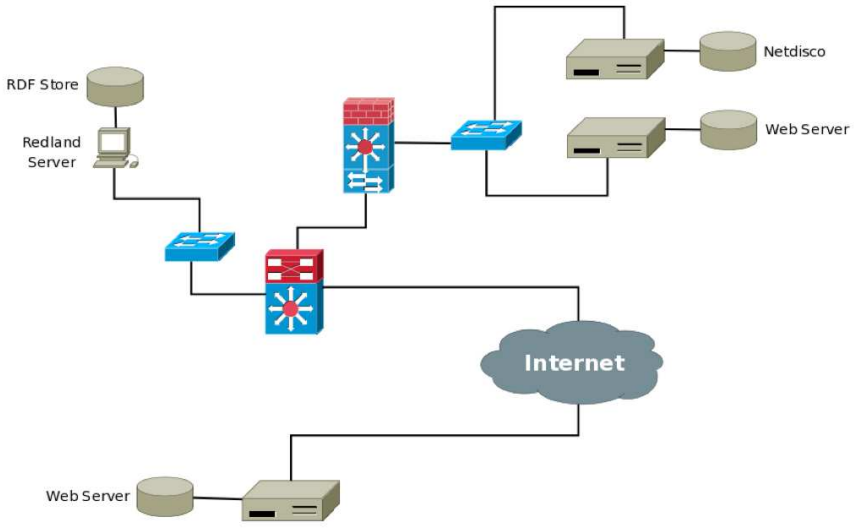

Figure 13. A modified version of Figure 1 that includes the Redland platform.

[13] These declarations correspond to those at the start of Listing 1.

Just to reiterate, and again in reference to Figure 8, the outcome at this point is a relationship-centric representation for the topology of the York University IP network. Furthermore, the relationships have been serialized as RDF/XML and are now available for other purposes. In the following section, the querying of the resulting `.rdf` file is given consideration, as this allows us to illustrate how the current approach is quite different from that employed by Netdisco.

3.3. Querying Relationship-Centric Representations

In this section, the assumption of a pre-existing relationship-centric representation in RDF/XML is made. Given this starting point, attention focuses on processing queries against the same. Thus Algorithm 3, save for expression of the query itself in SQL, is comprised of language-neutral pseudo-code. As before, the choice was made to make use of the Redland RDF Library [12], and specifically the Perl bindings for the same.[14] After enabling the Redland environment, various declaration statements are made. Included in these declarations is the name of the `.rdf` file that encapsulates the relationship-centric representation of the network-topology data.[15] As before (i.e., in Algorithm 2), a storage allocation is made to allow for the instantiation of an RDF-based model for the queries. To ingest the RDF/XML representation of the network-topology data for use within the platform, an instance of Redland's Raptor Parser is created (via `RDF::Redland::Parser`). Using the parser, the purpose of the first `while` loop of Algorithm 3 is to populate the newly created RDF model (and therefore the corresponding database instance) with statements derived from parsing the `.rdf` representation of the network-topology data; relevant namespaces are also detected during this parsing phase and stored to an array for later use.

Redland offers several mechanisms for query handling (via `RDF::Redland::Query`). Arguably the most appropriate of these in the context of the Semantic Web is that based on SPARQL [1]. With respect to Algorithm 3, it is assumed that the query requiring execution is contained

[14] Although Redland's Perl bindings were used again here, this algorithm could be implemented using, e.g., Redland's bindings for Python. This flexibility could be particularly advantageous when, for example, different developers or teams are involved.

[15] Of course, there are other options available for storing an RDF-based relationship-centric representation. As is clear from the previous section, relational databases are one such medium.

Algorithm 3 Pseudo-code skeleton for the RDF-querying algorithm (modeled after a usage example in Perl provided with Redland [12].)

```
{Invoke external packages - e.g., RDF platform}
{Define variables - including the name of the .rdf file}
{Open the .rdf file for input}
{Instantiate a persistent database for this RDF representation}
{Instantiate an RDF model for this representation in the persistent database}
{Instantiate RDF/XML parser}
while (there is data from the file to parse as a stream) do
   {Add parsed streams as statements to the RDF model}
   {Detect namespaces during parsing and store to an array}
end while
{Open the file containing the query in SPARQL for reading}
while (there are statements in the RDF model to search against as a stream)
do
   {Query each statement for search-term match}
   {Save matches in an array}
end while
{Instantiate RDF/XML serializer}
{Establish namespaces for RDF/XML serialization}
{Serialize query results to a .rdf file as output}
{Close all open files}
{Deconstruct the serializer}
{Deconstruct the persistent data store}
{Deconstruct the RDF model}
{Visualize results in the .rdf file graphically}
```

in a separate file (see, e.g., Listing 2). Thus this query file is opened prior to the execution of the second `while` loop of Algorithm 3. This latter loop executes for as long as it has statements to process from the RDF model of the network-topology data; in executing, the SPARQL-based query is applied to the statements from the RDF model, and matches are stored to an array. In a subsequent step, once Redland's serializer has been instantiated, the matches are serialized to RDF/XML (i.e., a .rdf file). The remainder of Algorithm 3 focuses

on closing open files, gracefully decommissioning the Redland environment, and visualizing the results. As was the case for Algorithm 2, visualization of the results is included in Algorithm 3 for completeness even though this is a step handled outside of the Redland platform.

In general, SPARQL queries are comprised of the following four elements [27]:

1. Prefix declarations — for abbreviating URIs;
2. A results clause — for identifying the information that needs to be returned by the query;
3. The query pattern — for specifying that which is being queried from the underlying dataset; and
4. Query modifiers — for slicing, ordering, and otherwise rearranging query results.

Each of these elements is in evidence in Listing 2. As expected, the Listing begins (lines `1-3`) by declaring the namespaces required to expand the references to full URIs made elsewhere in the query. Next, `CONSTRUCT` (lines `5-9`) is used to indicate the required results. In this case, for the identified `?subject`[16] of the RDF representation (i.e., the IP address(es) of the corresponding Cisco switch(es)), triples are sought for corresponding `nkm:neighbor`, `nkm:ip_name` and `nkm:ip_device_type` properties; for the first two properties, all values are sought, whereas the string literal `6509` is the explicitly required match in the case of the `nkm:ip_device_type`. In addition to these results, and for each match of the identified `nkm:neighbor` property by the `?object1` variable, Listing 2 also returns as results the corresponding `nkm:ip_name` and `nkm:ip_device_type`. Other options exist (e.g., `SELECT`) for specifying the results clause [69]. `CONSTRUCT` is used here as it returns results in the form of triples serialized as RDF/XML; and, as is shown shortly (Figure 14), the triples resulting from the query can then be visualized graphically.

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3 PREFIX nkm: <http://www.yorku.ca/cns/netops/nkm/1.0/>

5 CONSTRUCT { ?subject nkm:neighbor ?object1 ;
```

[16] Note that the `?` preceding `subject` indicates that `?subject` is a variable.

```
6                       nkm:ip_name ?object2 ;
7                       nkm:ip_device_type "6509" .
8           ?object1 nkm:ip_name    ?object3 ;
9                    nkm:ip_device_type ?object4 . }

11 WHERE { ?subject nkm:neighbor ?object1 ;
12                      nkm:ip_name ?object2 ;
13                      nkm:ip_device_type "6509" .
14          ?object1 nkm:ip_name    ?object3 ;
15                   nkm:ip_device_type ?object4 .

17 FILTER ( ?object4 = "6509" || ?object4 = "6506" ) .

19 }
```

Listing 2: SPARQL-based query to generate a network knowledge map of all Cisco 650x switches by IP name and address. The corresponding map is presented in Figure 14.

The `WHERE` clause (lines `11-15` of the Listing) is used to detail the query pattern. In the case of Listing 2, this query pattern closely resembles the results sought after via the `CONSTRUCT` clause. Again, identified predicates (i.e., `nkm:ip_name` and `nkm:ip_device_type`) are sought for the network neighbors resulting from the `?subject-nkm:neighbor-?object1` triple. And for each of these neighbors (i.e., the `object1`'s of the previous statement), again associated predicates (i.e., `nkm:ip_name` and `nkm:ip_device_type`) are queried for. In this latter case the value of `nkm:ip_device_type`, returned as variable `?object4`, is used as the basis for a query modifier. Specifically, the `FILTER` clause (line `17` of the Listing) is used to modify the query pattern so that only `?object4` matches with the string literals `6509` or `6506` are found.

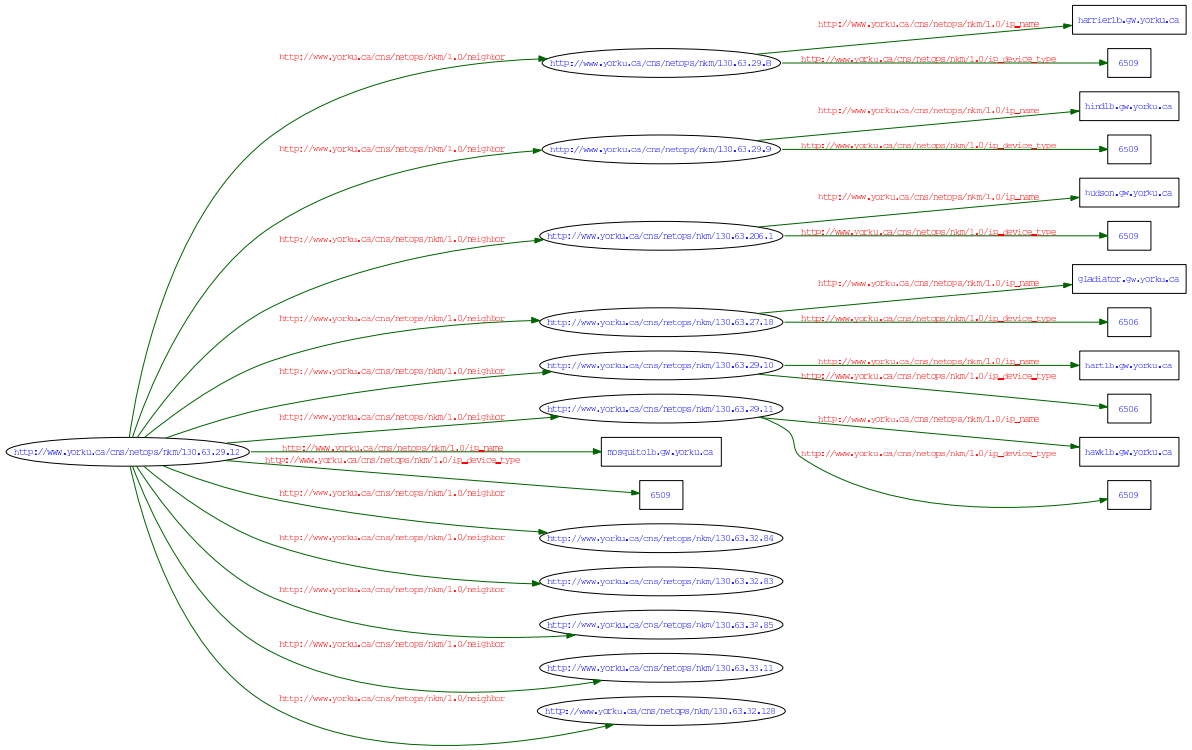

Figure 14. The network knowledge map corresponding to the SPARQL query presented in Listing 2.

It is readily evident from Listing 2 that SPARQL is a 'Webified' dialect of SQL. This 'Webification' pays dividends in the context of visualizing the results of the query contained in Listing 2. As alluded to above, use of the `CONSTRUCT` clause was deliberate as results are returned as triples that can be serialized as RDF/XML. Therefore the SPARQL query contained in Listing 2 was used as input for an implementation of Algorithm 3 using the Perl bindings for the Redland platform. Of course, along with the query, the RDF representation derived previously (see §3.2.) was also a required input. The outcome of applying the SPARQL query of Listing 2 to the previously derived relationship centric representation is the graphical representation provided in Figure 14.

From the outset, it is tempting to interpret Figure 14 along the lines of Figure 7 — i.e., as a map of the topology for the IP network core at York University. This interpretation is not entirely incorrect as Figure 14 certainly includes representations that are topological in nature. For example, it is certainly a representation of the Figure that the switch with IP address `130.63.206.1` is topologically a CDP neighbor (`nkm:neighbor`) of the switch `130.63.29.12`. In addition to presenting such topologies, Figure 14 also indicates that `hudson.gw.yorku.ca` and `6509` are the corresponding IP name (`nkm:ip_name`) and device type (`nkm:ip_device_type`), respectively, for the switch identified by the IP address `130.63.206.1`. Because each of the predicates `nkm:neighbor`, `nkm:ip_name` and `nkm:ip_device_type` were actually used in the SPARQL query, Figure 14 is properly the network knowledge map corresponding to the query of Listing 2. In other words, Figure 14 conveys knowledge based on the relationship-centric representation that is in excess of the purely topological representation available from Netdisco (e.g., Figure 7).

Figure 14 is readily intelligible to a human. Even purely textual representations along the lines of Listing 1 can be parsed by humans. This is the self-describing facet of XML that is therefore evident in RDF/XML. Of course, this is appealing, as it is useful that results derived can be interpreted by humans. However, this relatively simple investigation also demonstrates that these same relationship-centric models and results are also *known* to machines. In other words, the RDF-based approach taken here addresses the previously identified expressibility gap, as now that which is known to humans (the network specialist) is also known to machines.

Listing 2 is the SPARQL equivalent of the following expression in natural language:

> Illustrate graphically all Cisco 650x class switches connected to the core switch (Mosquito) at York University. For each switch, be sure to include its IP name, address and model type.

Other than technically interpreting "connected" as "is a CDP-neighbor of", Listing 2 captures the essence of the natural-language expression. Realized here, this need to query relationship-centric representations has been an established objective in a broader semantic framework for some time [31]. Thus, in the next section, the current effort is placed in this broader context.

4. A Semantic Framework for Relationship-Centric Representations

4.1. The Basic Framework

Netdisco's use of a relational database allowed data to be purposefully extracted and used to populate the relationship-centric representation developed in §3.1. In other cases in networking, and indeed in other disciplines, schema-oriented data is not available. In fact, files commonly serve as repositories of loosely to rigidly structured data in ASCII or binary formats. Motivated originally by use cases of scientific origin, a semantic framework has been developed to ultimately allow for the transformation of semi-structured ASCII data into knowledge-centric representations.[17] Since the details of the scientifically oriented use case are provided elsewhere [28–31], the visual summary presented in Figure 15 is described here from the following discipline-neutral, three-stage perspective:

Stage I To fix ideas, suppose `primary.dat` (in the Figure) is the ASCII data file. The approach taken here is to create an XML-based representation of `primary.dat` via transformation; the transformation operates on the

[17] With minimal effort, the framework developed here can be enhanced to support files in binary formats.

data in its legacy format and actually makes use of the legacy format.[18] To facilitate the transformation to an XML-based representation, templates are used to capture the specifics of the legacy data format. In numerous cases discipline-specific dialects of XML already exist (illustrated as `*ML` in the Figure), and in some of these cases, so does the corresponding XML schema (`*ML.xsd` in the Figure). Through use of an appropriate processor (`*ML Processor` in the Figure), the schema-based template (`*ML Template` in the Figure) allows for transformation of the input data into an XML-based representation (serialized in XML as `primary.xml` in the Figure). Although discipline-specific formalisms can be appealing,[19] the impending availability of an implementation of the standards-based Data Format Description Language (DFDL, [50]) is anticipated to be of broader applicability and therefore overall value.

Stage II From the XML-based representation (`primary.xml`) of the original data (`primary.dat`), relationships are automatically extracted. Critical to this second stage is the use of eXtensible Stylesheet Language Transformations (XSLT, [75]) rules (`primary_RDF.xsl` in the Figure) to facilitate the conversion and software (`GRDDL Processor` in the Figure) that produces RDF representations (serialized in RDF/XML as `primary.rdf` in the Figure) via Gleaning Resource Descriptions from Dialects of Languages (GRDDL, [18, 59, 60]) based processing.[20]

Stage III From the RDF-based representation of the data (`primary.rdf` in the Figure), classes, properties and individuals are automatically extracted (serialized in OWL/XML as `primary.owl` in the Figure). Although a strategy for this extraction of Web Ontology Language (OWL, [63]) from RDF does exist [65], the availability of actual implementations (`OWL Processor` in the Figure) requires further research. Given this implementation status, the seemingly reasonable allusion to XSLT rules (`primary_OWL.xsl` in the Figure) should be taken as speculation only at this time.

[18] Using this approach means that data in its legacy format remains the authoritative source, with the transformed data in XML being an on-demand derived product, or 'virtual XML' [42].

[19] See, for example, [28, 29] in regard to the scientific use case for Earth Sciences Markup Language (ESML, [16, 41]).

[20] It is noteworthy that GRDDL is a standard — a W3C Recommendation to be precise [72].

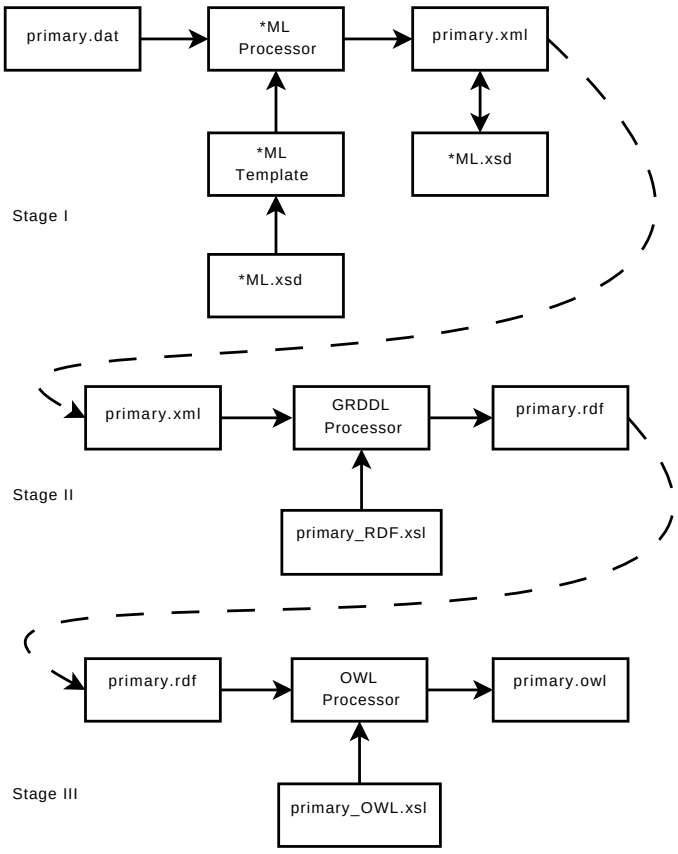

Figure 15. Schematic illustration of the progressive enhancement in semantic expressivity and richness of primary ASCII data. Semi-structured ASCII data is represented in XML via *ML, then RDF via GRDDL, and finally OWL.

In the current context of representing the topology of an IP network, Netdisco's use of a relational database completely eliminated the need for the first stage (i.e., `Stage I`) of transformations depicted in Figure 15. Moreover, the targeted queries directed at Netdisco's relational database allowed for rapid extraction of network-topology data to produce the relationship-centric representation in RDF discussed in §3.1., thus bypassing the need for use of GRDDL (i.e., `Stage II`). In certain respects, fast-tracked 'RDFization' is not surprising, as

the schema-oriented relational database provides a vantage point that is quite simply absent in the case of semi-structured ASCII data. Unfortunately, given the implementation status of the third stage of the framework (`Stage III` in the Figure), no attempt was made here to extract an OWL-based representation of the network-topology data. Taken against the backdrop of Figure 15, the current case involving network-topology data also emphasizes why use of the word *framework* is essential in the current context. In other words, Figure 15 is intended to provide an over-arching context and strategy for achieving systematically more expressive representations of data; it is not intended to provide a algorithm that must be strictly adhered to.

In an automated, bottom-up process, the systematic enhancement in semantic expressivity and richness (Figure 15) begins with a data representation in XML, and ultimately results in a knowledge representation in OWL. The resulting knowledge representation is referred to as an *informal ontology* [29].[21] Because they can in principle be constructed on demand, these data and knowledge representations have also been referred to as virtual XML [42] and virtual ontologies [30], respectively.

The semantic framework presented here serves well in representing primary data, i.e., `primary.dat` in Figure 15. Again, motivated by the scientific use case alluded to above, there is a need to represent aspects or features of the primary data.[22] Thus the following section describes how the semantic framework has been extended to incorporate additional information on the primary data as annotations.

4.2. Annotations and the Semantic Framework

Using Amaya [58],[23] an RDF comment (`<rdf:comment>`) was added to the RDF/XML serialization developed previously. Shown here in the lower-left panels of Figures 16 and 17 is this comment as the string: `This is`

[21] Even though they are both data-oriented, informal ontologies are separate and distinct from the tagging-based social networking representation known as folksonomies [34, 44].

[22] In the scientific use case, the features requiring representation were recorded as an event log [31]. Specifically, this event log allowed instrument operators to provide additional information on the primary data that was being collected.

[23] Amaya is a Web client from the W3C that acts both as a browser and as an authoring tool [58].

`one of the legacy devices in the network core.` By visually parsing the RDF/XML representation of Figure 16 or the structural representation of Figure 17, the `This` of the comment string is identifiable as the subject `130.63.29.3`; it is also evident that `130.63.29.3` has an IP name of `oxfordlb.gw.yorku.ca` and IP device type of `WSX5302`. Based on examples such as this one, it is clear that the additional information alluded to at the close of the previous section, can be incorporated within the relationship-centric representation - i.e., internally to the `.rdf` file produced in the second stage of the semantic framework (Figure 15).

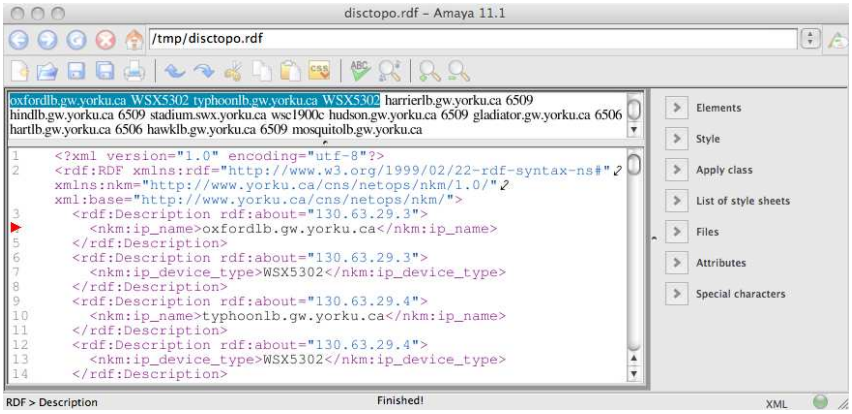

Figure 16. A screenshot of Amaya (Version 11.1) showing rendered RDF/XML in the upper-left panel and source RDF/XML in the lower-left; RDF/XML elements are displayed on the right.

By selecting the string `oxfordlb.gw.yorku.ca WSX5302 typhoonlb.gw.yorku.ca WSX5302` in the upper-left panel of either Figure 16 or 17, and then `Tools | Annotations | Annotate Selection`, it is also possible to enter additional information via the annotation window illustrated here in Figure 18. In addition to the string itself (i.e., `These are some of the legacy devices in the network core.`), the title, author, source document, annotation type plus creation and modification dates collectively comprise *metadata* regarding the additional information. In striking contrast to the `<rdf:comment>` introduced above, this *annotation* (or editorial metadata) is *external* to the `.rdf`

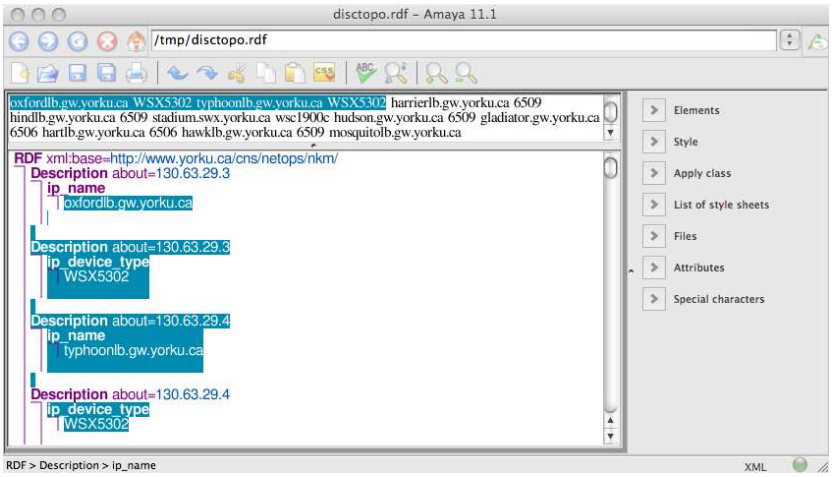

Figure 17. A screenshot of Amaya (Version 11.1) showing rendered RDF/XML in the upper-left panel and structured RDF/XML in the lower-left; RDF/XML elements are displayed on the right.

file that contains the relationship-centric representation; in fact, the RDF/XML serialization of the annotation (Listing 3 and Figure 19) can be stored locally or remotely.[24] Included in the namespace declarations of Listing 3 is the Dublin Core [49] (line `6`) plus one that specifically relates to annotations (line `3`).[25] This latter declaration allows `Advice` to be identified as the annotation type (line `9`) for the target URL, namely `file:///tmp/disctopo.rdf`. The range selected for annotation (line `11`) is detailed through use of the location types (`start-point()` and `end-point()`) and functions (`string-range()` and `range-to()`) from XML Pointer Language (XPointer, [74]).[26] Even a cursory examination of the arguments of the

[24]For a more-comprehensive discussion of annotations via Amaya consult, e.g., [31]. Annozilla [35], an alternative to Amaya, has also been investigated in a related context [30].

[25]Note also that Amaya declares but does not actually use some of the namespaces appearing in Listing 3.

[26]The standard for XPointer is comprised of four W3C specifications. Specifications for full XPath-based addressing [26], element-position addressing [23], and namespace binding [25], are each cast within a framework [24]. Examination of [26] reveals how XPointer numbers nodes and points, [22], to provide the relative addressing alluded to above.

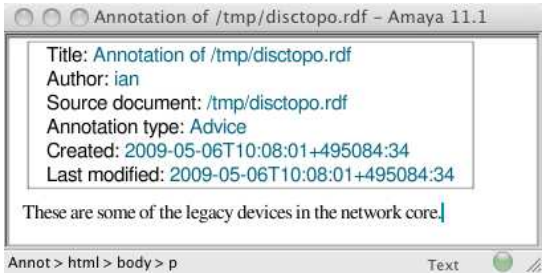

Figure 18. The Amaya annotation window resulting from invocation of `Tools | Annotations | Annotate Selection`.

`string-range()` function makes it clear that XPointer allows for URI fragment identification via a scheme that respects the XML Document Object Model (DOM, [70]). Illustrated graphically via Figure 17, it is also clear that the extensibility of XML is preserved in the context of the DOM and XPointer, as elements (e.g., `<nkm:ip_device_type>`) specific to, e.g., the `http://www.yorku.ca/cns/netops/nkm/1.0/` namespace are treated appropriately. XPointer's ability to serve as a URI fragment identifier also means that annotations can cross-cut RDF/XML element boundaries; thus the subtle change in wording (i.e., from `This` to `These`) is a reflection of an anything but subtle change in context (i.e., from subject `130.63.29.3` to the URI fragment shown as selected in either of Figures 16 or 17). In other words, as a URI fragment identifier, annotations can have a scope that is much broader than that available to an `<rdf:comment>`. Lines `12-15` of Listing 3 detail metadata evident in Figure 18. Note that the content (i.e., the editorial metadata) of the annotation is itself contained within an XHTML [21] file (Listing 4) as indicated on line `16` of Listing 3.

```
1 <?xml version="1.0" ?>
2 <r:RDF xmlns:r="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3  xmlns:a="http://www.w3.org/2000/10/annotation-ns#"
4  xmlns:t="http://www.w3.org/2001/03/thread#"
5  xmlns:http="http://www.w3.org/1999/xx/http#"
6  xmlns:d="http://purl.org/dc/elements/1.1/">
7 <r:Description>
8 <r:type r:resource="http://www.w3.org/2000/10/annotation-ns#↘
  →Annotation" />
```

```
<r:type r:resource="http://www.w3.org/2000/10/annotationType#Advice" />
<a:annotates r:resource="file:///tmp/disctopo.rdf" />
<a:context>file:///tmp/disctopo.rdf#xpointer(start-point(string-range(/RDF[1]/Description[1]/ip_name[1],"",1,1))/range-to(end-point(string-range(/RDF[1]/Description[4]/ip_device_type[1],"",7,1))))</a:context>
<d:title>Annotation of /tmp/disctopo.rdf</d:title>
<d:creator>ian</d:creator>
<a:created>2009-05-06T10:08:01+495084:34</a:created>
<d:date>2009-05-06T10:09:08+495084:34</d:date>
<a:body r:resource="file:///Users/ian/Library/Application%20Support/amaya/annotations/annotWPxsvx.html" />
</r:Description>
</r:RDF>
```

Listing 3: Amaya's XPointer-based specification of the annotation depicted in Figure 18.

Figure 19 is obtained by parsing Listing 3 with the W3C's RDF Validation Service [71] and graphing the resulting subject-predicate-object triples. Because no subject has been identified, the parser assigns a generic identifier (`genid:A23036`). With this assignment, the remainder of the graph states the relationships between this subject resulting from parsing Listing 3. The effective use of various namespaces is clearly evident from this graphical approach to visualizing RDF triples. Note also in Figure 19 that some predicates are themselves URIs whereas others are string literals. Finally, Figure 19 makes graphically and syntactically explicit an annotation via XPointer.

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Annotation of /tmp/disctopo.rdf</title>
</head>
<body>
<p>These are some of the legacy devices in the network core.</p>
</body>
</html>
```

Listing 4: XHTML content (i.e., the editorial metadata itself) of the Amaya annotation depicted in Figure 18.

Figure 19. Graphical visualization resulting from parsing Listing 3 by the W3C's RDF Validation Service [71].

Amaya's ability to allow for XPointer-based annotations is appealing on a number of levels. The serialization in RDF/XML (e.g., Listing 3), in tandem with graph-oriented RDF's inherent ability for merging independent representations [1], results in the unsurprising outcome that XPointer-based annotations have been incorporated into the semantic framework [31, §4]. Subsequently, however, in an annotation-modeling exercise [30], it was concluded:

> OWL possesses an internal mechanism for annotation that is constrained to ensure computational completeness and decidability. This allows for all-in-one annotations and ontologies. Although all-in-one representations have inherent appeal, there are cases where externally originating annotations based on the XML Pointer Language (XPointer) are preferable or even required. Unfortunately, XPointer-based annotations can easily violate the constraints that apply to OWL's internally based annotations. Constraint-violating annotations transform OWL DL into OWL Full.[27] All-in-one annotations and ontologies are still a possibility, however, such a representation is ill advised if computational completeness and decidability are desirable. Fortunately, the dynamic nature of annotations and informal ontologies, allows for decisions in real time and/or multiple outcomes.

Even though the above pertains specifically to OWL, this conclusion was arrived at through application of the annotation-enhanced semantic framework described here, i.e., in §4.1. In other words, the internal-to-the RDF/XML representation `<rdf:comment>` (e.g., along the lines illustrated in Figures 16 or 17), is constrained to be ultimately non-impacting for the computational completeness and decidability of the corresponding ontology. The same, however, cannot necessarily be said of the XPointer-based annotation (e.g., along the lines of Figure 18 plus Listings 3 and 4).

[27] OWL Description Logic (OWL DL) is appealing because it [46]: "... provides maximum expressiveness while ensuring computational completeness (all valid conclusions can be inferred) and decidability (the inferences take finite time)." Through the use of profiles, OWL 2 allows for an even greater degree of application centricity in terms of computational completeness and decideability. Thus it is anticipated that the OWL 2 profile specifically designed with relational databases in mind (i.e., OWL 2 QL, [62, §10.2]) will be of direct interest as a schema-based RDF representation (RDFS, [67]) for IP network knowledge is developed in §5.

5. Towards Schema-Based Relationship Centric Representations

5.1. Towards a Vocabulary for IP Network Knowledge

Because multidimensional arrays ultimately represented time-sampled data, for their scientific use case Lumb & Aldridge [29] concluded:

> ... early introduction of RDF should permit a better opportunity to ... Shape the specifics of the RDF representation — a capability whose value increases with the complexity of the XML representation, and/or when there exist multiple possibilities for extracting RDF from XML ... As such, it would desirable to shape the RDF representation in terms of RDF collections (e.g., [39, Chapter 4]) rather than explicit subject-predicate-object representations for each data point!

In an analogous fashion, it was also suggested that a similar approach might prove useful in organizing annotations [31].[28] Even though time series are not involved in the current context of knowledge maps for IP networks, it is clear that there are various cases in evidence in which it makes sense to organize via RDF collections or containers. For example, such a scheme would permit reference to end-of-life network switches en masse.

RDF collections and containers are not the only means for introducing organization. In fact, an approach that is semantically more expressive is initiated by introducing sets. Using RDF Schema (RDFS, [67]), Listing 5 introduces the concept of a `Cisco:Product` set via the `rdfs:Class` construct (line `3`).[29] (Note that `Cisco` is the namespace prefix identified on the first line of the Listing; namespace prefixes for RDF and RDFS appear on the following two lines respectively.) Lines `4-7` of the Listing successively identify specific Cisco Catalyst Series switches as *asserted* set members of the Cisco product portfolio via

[28] Of course, based on concerns raised elsewhere [30] and here in §4.2., the resulting group of annotations may have undesirable characteristics around computational decideability and completeness.

[29] Following [1], and for readability, RDF subject-predicate-object triples are serialized in this section via N3 notation [61] rather than RDF/XML.

the `rdfs:subClassOf` predicate. And finally, the Cisco Catalyst 5302 is asserted as a product within (i.e., `rdfs:subClassOf`) both the Cisco Catalyst 5000 and 5500 Series product portfolios.

```
1 @prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix Cisco: <http://www.yorku.ca/cns/netops/vendors/rdfs/↘
  →Cisco/> .

5 Cisco:Product       rdf:type rdfs:Class .
6 Cisco:Catalyst4500 rdfs:subClassOf Cisco:Product .
7 Cisco:Catalyst6500 rdfs:subClassOf Cisco:Product .
8 Cisco:Catalyst5000 rdfs:subClassOf Cisco:Product .
9 Cisco:Catalyst5500 rdfs:subClassOf Cisco:Product .
10 Cisco:Catalyst5302 rdfs:subClassOf Cisco:Catalyst5000 .
11 Cisco:Catalyst5302 rdfs:subClassOf Cisco:Catalyst5500 .
```

Listing 5: RDFS definition of the `Cisco:Product` class and some of its subclasses.

To further develop the application of RDFS in this context, consider now the four, asserted properties (respectively starting on lines `1`, `4`, `7` and `10`) identified in Listing 6. Originating from the vendor's end-of-life-milestones [8, Table 1], each of these properties is declared as being a date string (i.e., `rdfs:Literal`) of type (i.e., `rdf:type`) `rdf:Property`. Through use of `rdfs:domain`, it is also made explicit that each of these properties applies to the `Cisco:Product` class.[30] By inference then, these properties also apply to all subclasses of the `Cisco:Product` class - e.g., to `Cisco:Catalyst5302`.

```
1 Cisco:EndOfLifeAnnouncementDate rdf:type     rdf:Property
2                                 rdfs:range   rdfs:Literal
3                                 rdfs:domain Cisco:Product .
4 Cisco:EndOfSaleDate             rdf:type     rdf:Property
5                                 rdfs:range   rdfs:Literal
6                                 rdfs:domain Cisco:Product .
7 Cisco:LastDateOfSupportHardware rdf:type     rdf:Property
8                                 rdfs:range   rdfs:Literal
9                                 rdfs:domain Cisco:Product .
10 Cisco:LastDateOfSupportSoftware rdf:type     rdf:Property
```

[30] The recontextualization of mathematical terms such as set, domain, and range, is intentional and meaning rich in RDFS [1].

```
11                                  rdfs:range   rdfs:Literal
12                                  rdfs:domain  Cisco:Product .
```

Listing 6: Extracted RDFS property definitions relating to the `Cisco:Product` class and its subclasses.

Using the schema developed thus far in Listings 5 and 6, the instances `Cisco:Catalyst5000_1` and `Cisco:Catalyst5500_1`[31] are shared via the `.rdf` file presented in Listing 7. The instances include actual dates from the vendor's end-of-life announcement [8, Table 1]. Because `Cisco:Catalyst5302` is a subclass of the classes involved in Listing 7, any instance of this subclass (e.g., `Cisco:Catalyst5302_1`) is beholden to the *same* end-of-life criteria by *inference*.

```
1  @prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2  @prefix Cisco: <http://www.yorku.ca/cns/netops/vendors/rdfs/↘
   →Cisco/> .

4  Cisco:Catalyst5000_1
5        Cisco:EndOfLifeAnnouncementDate "2002-10-29T00:00:01↘
         →+000000:00"
6        Cisco:EndOfSaleDate             "2003-06-30T23:59:59↘
         →+000000:00"
7        Cisco:LastDateOfSupportHardware "2008-06-30T23:59:59↘
         →+000000:00"
8        Cisco:LastDateOfSupportSoftware "2006-06-30T23:59:59↘
         →+000000:00" .

10 Cisco:Catalyst5500_1
11       Cisco:EndOfLifeAnnouncementDate "2002-10-29T00:00:01↘
         →+000000:00"
12       Cisco:EndOfSaleDate             "2003-06-30T23:59:59↘
         →+000000:00"
13       Cisco:LastDateOfSupportHardware "2008-06-30T23:59:59↘
         →+000000:00"
```

[31] `_1` has been appended to the class names to distinguish the instance from the class itself. In OWL, instances of classes are also referred to as individuals [64, Section 5].

```
14      Cisco:LastDateOfSupportSoftware "2006-06-30T23:59:59↘
        →+000000:00" .
```

Listing 7: RDF definition of `Cisco:Catalyst5000_1` and `Cisco:Catalyst5500_1` class instances with vendor-specific data.

Listing 8 reiterates two relationships evident in Figures 16 and 17. By way of their IP addresses, `oxfordlb.gw.yorku.ca` and `typhoonlb.gw.yorku.ca` are declared as being of type (i.e., `rdf:type`) `Cisco:Catalyst5302`. Therefore, and again by inference, it is *known* that these York University routers [10] are also beholden to the same end-of-life criteria detailed in the schema comprised of Listings 5 and 6.

```
1 @prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix Cisco: <http://www.yorku.ca/cns/netops/vendors/rdfs/↘
  →Cisco/> .
3 @prefix nkm:   <http://www.yorku.ca/cns/netops/nkm/1.0/> .

5 130.63.29.3 rdf:type    Cisco:Catalyst5302
6             nkm:ip_name oxfordlb.gw.yorku.ca .
7 130.63.29.4 rdf:type    Cisco:Catalyst5302
8             nkm:ip_name typhoonlb.gw.yorku.ca .
```

Listing 8: Asserted RDF triples for two Cisco Catalyst 5302 switches at York University.

Even with only the underpinnings of an RDFS-based vocabulary for networking (Listings 5 and 6), it is undeniably evident that *inferencing* is where the value of RDFS really is delivered [1]. It is important to state that inferencing also applies in the context of querying relationship-centric representations (cf. §3.3.). In other words, SPARQL-based queries (e.g., Listing 2) will return both asserted *and* inferred triples matching query criteria.

5.2. Automating Development of Vocabularies for IP Network Knowledge

As illustrated in the previous section (§5.1.), it is possible to manually enhance a relationship-centric representation through the addition of a schema (i.e., Listings 5 and 6). And when that schema is based on RDFS, the inherent capability for inferencing affords fully expressive asserted and inferred relationships from the perspective of representation. This allowed, as an example, a very efficient

mechanism for propagating a vendor's end-of-life announcement to a certain class of switches, and application of the same to specific switches in the York University IP network. Though not explored here,[32] such representations are also amenable to queries — queries that have the potential to return in response both asserted and inferred relationships. Though feasible, the manual development of RDFS-based schemas is a somewhat tedious undertaking that has the potential to impede progress. Thus, in the remainder of this section, attention focuses on prospects for automating schema development.

As Listing 9 explicitly demonstrates, Integrated Development Environments (IDEs) such as Protégé [40] are particularly effective in this automation context. Not only do such IDEs accelerate schema development, they allow for the production of RDFS-based schemas that validate correctly. In other words, it is possible to use Listing 9 as input for the W3C RDF Validation Service [71], and receive as output the resulting RDF triples and a graphical representation of the same.[33] Protégé-plugin OntoViz [33] also allows for visualization of RDFS-based schemas within the context of the IDE (Figure 20). Note in the Figure that in addition to the class hierarchy, class properties are displayed. Clearly Protégé offers some possibilities for automating schema development. At some point, however, classes and properties need to be manually entered into Protégé. Again, the question that needs to be asked is if this process of schema development can be automated.

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <!DOCTYPE rdf:RDF [
3   <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
4   <!ENTITY Cisco 'http://www.yorku.ca/cns/netops/vendors/rdfs/↘
    →Cisco:'>
5   <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
6 ]>
7 <rdf:RDF xmlns:rdf="&rdf;"
8          xmlns:Cisco="&Cisco;"
9          xmlns:rdfs="&rdfs;">
10 <rdfs:Class rdf:about="&Cisco;Catalyst4500"
11          rdfs:label="Catalyst4500">
12        <rdfs:subClassOf rdf:resource="&Cisco;Product"/>
13 </rdfs:Class>
```

[32] SPARQL-based queries of RDFS-enhanced relationship centric representations is a topic of current research. Particular attention is being paid to inferred relationships.

[33] Such graphical representations have been shared here previously — see, e.g., Figure 11.

```
<rdfs:Class rdf:about="&Cisco;Catalyst5000"
          rdfs:label="Catalyst5000">
        <rdfs:subClassOf rdf:resource="&Cisco;Product"/>
</rdfs:Class>
<rdfs:Class rdf:about="&Cisco;Catalyst5500"
          rdfs:label="Catalyst5500">
        <rdfs:subClassOf rdf:resource="&Cisco;Product"/>
</rdfs:Class>
<rdfs:Class rdf:about="&Cisco;Catalyst6500"
          rdfs:label="Catalyst6500">
        <rdfs:subClassOf rdf:resource="&Cisco;Product"/>
</rdfs:Class>
<rdfs:Class rdf:about="&Cisco;CatalystWSX5302"
          rdfs:label="CatalystWSX5302">
        <rdfs:subClassOf rdf:resource="&Cisco;Catalyst5000"/>
        <rdfs:subClassOf rdf:resource="&Cisco;Catalyst5500"/>
</rdfs:Class>
<rdf:Property rdf:about="&Cisco;EndOfLifeAnnouncementDate"
          rdfs:label="EndOfLifeAnnouncementDate">
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:domain rdf:resource="&Cisco;Product"/>
</rdf:Property>
<rdf:Property rdf:about="&Cisco;EndOfSaleDate"
          rdfs:label="EndOfSaleDate">
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:domain rdf:resource="&Cisco;Product"/>
</rdf:Property>
<rdf:Property rdf:about="&Cisco;LastDateOfSupportHardware"
          rdfs:label="LastDateOfSupportHardware">
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:domain rdf:resource="&Cisco;Product"/>
</rdf:Property>
<rdf:Property rdf:about="&Cisco;LastDateOfSupportSoftware"
          rdfs:label="LastDateOfSupportSoftware">
        <rdfs:range rdf:resource="&rdfs;Literal"/>
        <rdfs:domain rdf:resource="&Cisco;Product"/>
</rdf:Property>
<rdfs:Class rdf:about="&Cisco;Product"
          rdfs:label="Product">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
</rdf:RDF>
```

Listing 9: Protege-produced RDFS definition for the `Cisco:Product` class,

subclasses and properties.

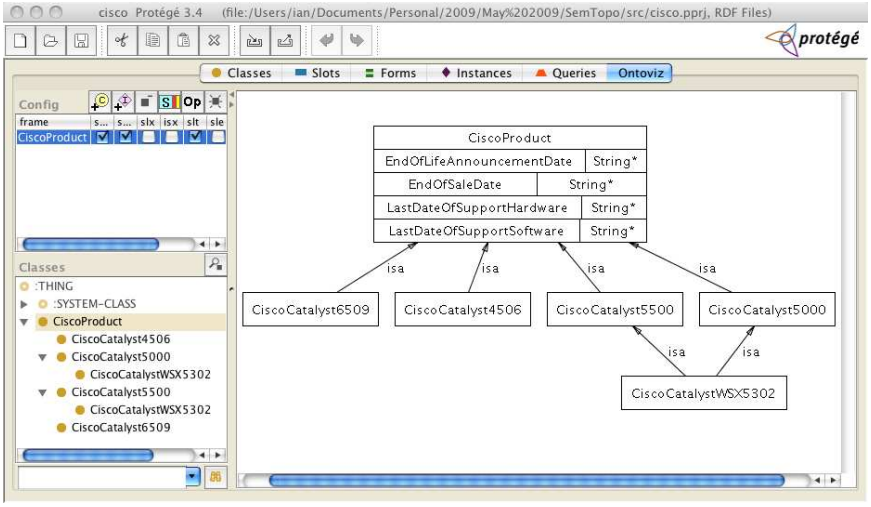

Figure 20. Graphical visualization of the RDFS-based schema for the `Cisco:Product` class and its properties via Protégé's OntoViz plugin.

```
moduleEntry       OBJECT-TYPE
  SYNTAX              ModuleEntry
  MAX-ACCESS    not-accessible
  STATUS              current
  DESCRIPTION   "Entry containing information about one module in
                          the chassis."
  INDEX               { moduleIndex }
  ::= { moduleTable 1 }

ModuleEntry ::=
  SEQUENCE {
     ...
     moduleType
       INTEGER,
     ...
     moduleSubType
     INTEGER,
     ...
```

```
19   }

21 moduleType OBJECT-TYPE
22   SYNTAX        INTEGER {
23      ...
24      wsx5302(52), -- Vlan router
25      ...
26      }
27      MAX-ACCESS   read-only
28      STATUS              current
29      DESCRIPTION   "The type of module."
30      ::= { moduleEntry 2 }

32 moduleSubType OBJECT-TYPE
33   SYNTAX              INTEGER {
34      ...
35      -- sub modules for the WS-X5302
36      wsx5304(6),  -- VIP II carrier
37      ...
38      }
39      MAX-ACCESS   read-only
40      STATUS              current
41      DESCRIPTION   "The type of daughterboard attached to this
42                                                  module."
43   ::= { moduleEntry 16 }
```

Listing 10: Manual extract of the CISCO-STACK-MIB from [9].

Another prospect for automating the development of an RDFS-based schema for IP network knowledge is hinted at through the following quote from [38, Chapter 1]:

> MIBs are specifications containing definitions of management information so that networked systems can be remotely monitored, configured and controlled.

Sandwiched between prose descriptions and references to other documents, these management-information definitions are cast as one or more MIB modules.[34] Articulated in an adapted subset of Abstract Syntax Notation One

[34] With respect to [38, Chapter 1], it is acknowledged that the term MIB is used somewhat loosely here to refer to a single specification (i.e., *a MIB*) or collection of specifications (i.e., *the MIB*). In the current context, however, such impreciseness does not compromise the generality or applicability of the current approach.

(ASN.1 [38, Chapter 3]) known as Structure of Management Information (SMI, [23]), an excerpt that demonstrates the utility of MIBs in the current context is shared via Listing 10. Although a detailed discussion of MIBs is deferred elsewhere (e.g., [38]), current purposes will be served by noting in Listing 10 that the `OBJECT-TYPE` construct is [38, Chapter 3]: "... either a class of management information or a mechanism to organize related object types." In the Listing `moduleEntry` (line `1`), `moduleType` (line `21`), and `moduleSubType` (line `32`), comprise three examples of this construct. Within the `moduleType` and `moduleSubType` object types, the RSM (`Vlan router`) is defined via the string `wsx5302` (line `24`) and the `VIP II carrier` submodule for the `wsx5302` via `wsx5304` (line `36`), respectively.[35] The trailing `(52)` and `(6)` afford registration of unique object identifiers (OIDs, [38, Chapter 2]) for the `wsx5302` and `wsx5304`, respectively. Finally with respect to Listing 10, it is noted that the parent nodes for `moduleType` and `moduleSubType` are identified as `moduleEntry` on lines `30` and `43`, respectively.

```
1 @prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix Cisco: <http://www.yorku.ca/cns/netops/vendors/rdfs/↘
  →Cisco/> .

5 Cisco:ModuleEntry    rdf:type rdfs:Class .
6 Cisco:ModuleType     rdfs:subClassOf Cisco:ModuleEntry .
7 Cisco:ModuleSubType  rdfs:subClassOf Cisco:ModuleType .

9 Cisco:WSX5302        rdfs:subClassOf Cisco:ModuleType .
10 Cisco:WSX5304       rdfs:subClassOf Cisco:ModuleType .
```

Listing 11: RDFS definition of the `Cisco:Product` class and some of its subclasses.

SNMP MIB's use of the `OBJECT-TYPE` construct affords manual deconstruction of Listing 10 and manual assembly of the RDFS-based schema of Listing 11. After defining the required namespaces, an RDFS-based class hierarchy is defined to match that manually extracted by parsing the MIB for object-type constructs and definitions within the same. The similarity between Listings 5 and 11 is also important to note, and should be taken as further validation of the viability of the approach proposed here. Finally, note that Listing 11 is a

[35] In MIBs, `--` are used to indicate comments.

succinct and comprehensive expression of certain Cisco products inspired from the MIB perspective. The choice of ASN.1-based SMI in the case of SNMP MIB modules, however, is known to have issues around expressiveness, for example [38, Chapter 3]:

> SNMP MIB modules are meant to be read by people as well as be read by programs. These programs are called MIB compilers. The result of having a specification meant for both people and programs is that the specification is difficult for people to read and programs to parse and use.

The above concern is readily evident in Listing 10. For example, for a human to parse the relationship between the `wsx5302` and `wsx5304` components, the class hierarchy (expressed via object types) needs to be deconstructed and all comments processed in context.

Despite concerns regarding the expressiveness in the use of SMI, there remains a degree of optimism for extracting an RDFS-based schema from SNMP MIBs. To that end, an outline of an automated process (adapted from §4.) for the same is illustrated schematically in Figure 21 and outlined as follows:

Stage I The same, vendor-supplied MIB (i.e., Listing 10 and `Cisco_MIB.mib` in the Figure) used by Netdisco comprises the input data. Based on this input, an XML-based representation of this MIB is produced (`Cisco_MIB.xml` in the Figure). The transformation from ASCII-based 'MIB format' (i.e., SMI) to XML is achieved via some to-be-developed processor,[36] template and XML Schema.[37]

Stage II Using XSLT transformation rules (`Cisco_MIB_RDF.xsl` in the Figure) as necessary, an RDFS-based representation of the MIB (`Cisco_MIB.rdfs`) is extracted from the XML-based representation with the aid of a GRDDL-based processor. In contrast to the previous application of the framework (§4.), the current application is schema-oriented — i.e., it is *not* instance-oriented, nor is it purely RDF. Thus,

[36] In addition to the `SNMP::Info` Perl module discussed in the context of Netdisco (§2.), there are a number of possibilities available for manipulating SNMP MIBs. Thus, there are quite likely existing prospects available for the `*ML Processor` identified in the Figure. This topic is discussed in greater detail in §6.

[37] As discussed in §6., components of this schema are already under development.

the outcome from application of this Stage of the semantic framework is an RDFS-based representation along the lines of Listing 9.

Stage III In this final Stage, there emerges the prospect of crafting an ontological (i.e., OWL-based, `Cisco_MIB.owl` in the Figure) representation of the MIB. Although there are some identified gaps in the extraction of OWL, as noted previously (§4.), there is cause for optimism as RDFS (and RDFS-Plus, [1, Chapter 7]) are syntactically and semantically 'closer' to OWL. Such a representation will again likely require rules-based processing to accomplish the transformation.

Though not explicitly drawn out in the Figure, IDEs like Protégé are likely to be of use in refactoring and refining the resulting schemas (Stage II) and ontologies (Stage III). Although it is the current contention that application of the semantic framework (as outlined above and illustrated in Figure 21) has the potential to extract relationship-centric representations from SNMP MIBs, further research is underway to validate this approach. As is demonstrated in the following section (§6.), others perceive significant value in such an extraction, and have even proposed related approaches.

6. Related Work

Broadly speaking, this chapter has been concerned with the intersection of IP network management (including discovery and monitoring) and knowledge representation. Slightly more specifically, this intersection has focused on SNMP and the Semantic Web, respectively. Even with this somewhat more limited scope, there is a significant corpus targeting each of these content areas in isolation. Therefore, in what follows here, effort is made only to review related work at the intersection of these areas.

Arguably, most impactful in this context, is a relatively recent undertaking by the IETF to provide the specifics for an XML-based representation of SNMP MIBs. To date, the most tangible outcome is an XML Schema for SNMP SMI datatypes [2]. As the `IpAddress` extract presented in Listing 12 illustrates, this IETF XML Schema will be of direct use in representing MIBs in XML during `Stage I` of the expressivity-enrichment process (Figure 21). Additionally, the restricted, patterned string definition of `IpAddress` in Listing 12

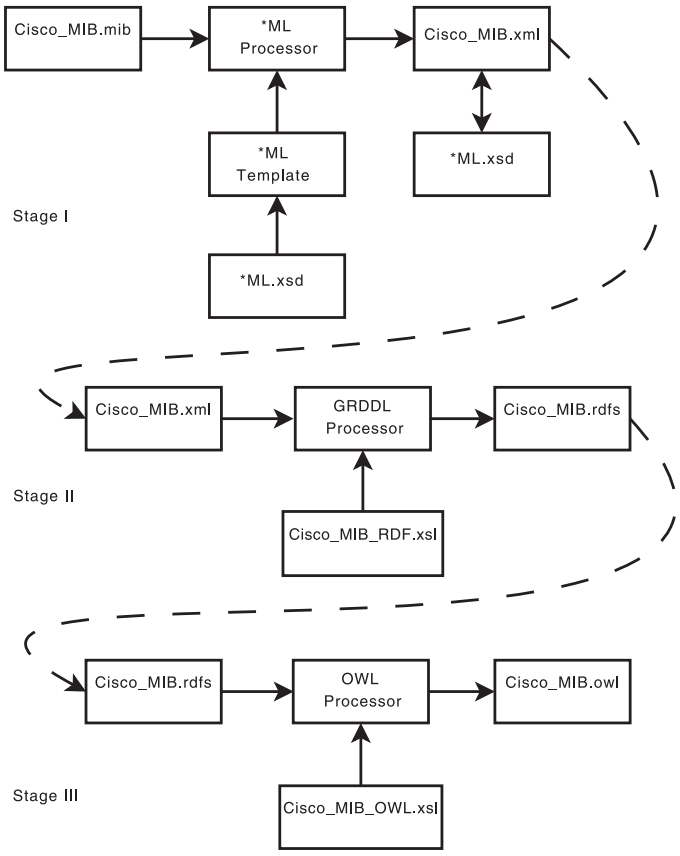

Figure 21. Schematic illustration of the progressive enhancement in Cisco_MIB.mib). Structured ASCII data (i.e., in SMI format) is represented successively in XML via *ML, then RDFS via GRDDL, and finally OWL.

could replace the nondescript string version of IP address starting in §3.1.

```
1 <?xml version="1.0" encoding="utf-8"?>
2   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3   xmlns="urn:ietf:params:xml:ns:opsawg:smi:base:1.0"
4   targetNamespace="urn:ietf:params:xml:ns:opsawg:smi:base:1.0↘
    →"
5   elementFormDefault="qualified"
```

```
6     attributeFormDefault="unqualified"
7     xml:lang="en">

9  ...

11 <xs:simpleType name="IpAddress">
12         <xs:restriction base="xs:string">
13           <xs:pattern value=
14           "((0|(1[0-9]{0,2})|
15 ␣␣␣␣␣␣␣␣␣␣␣␣(2(([0-4][0-9]?)|(5[0-5]?)|([6-9]?)))|
16 ␣␣␣␣␣␣␣␣␣␣␣␣([3-9][0-9]?))\.){3}
17 ␣␣␣␣␣␣␣␣␣␣␣␣(0|(1[0-9]{0,2})|
18 ␣␣␣␣␣␣␣␣␣␣␣␣(2(([0-4][0-9]?)|(5[0-5]?)|([6-9]?)))|
19 ␣␣␣␣␣␣␣␣␣␣␣␣([3-9][0-9]?))"/>
20         </xs:restriction>
21       </xs:simpleType>
```

Listing 12: XML Schema extract for the SNMP SMI `IpAddress` datatype from [2].

As an IETF 'work in progress', this introduction of an XML-based schema for SNMP SMI datatypes comprises a necessary step. In and of itself, however, this schema is insufficient in guaranteeing further progress; of course, this is known and acknowledged in various ways. In the current context, it is also important to convey that the original impetus [3] for developing an XML-based representation for SNMP MIBs came from anticipated, future requirements emerging in the areas of Web Services [73] and Service Oriented Architectures (SOAs, e.g., [14]). Also noteworthy, in the original ruminations on this topic within the IETF, was a cursory allusion to RDF [3]; no such allusion was made to RDFS (cf. §5.), however.

Closely related, though not officially IETF sponsored, are a number of efforts to develop SNMP-XML gateways [24,36,76]. Positioned as demonstrative examples, these gateways take significant steps beyond schema-based representations in XML of SNMP SMI datatypes by delivering working implementations. In principle, these gateways accept MIBs written in their native SMI format and return a schema-based representation in XML. On the topic of these gateways, Natale [3] concluded:

> ... that a large chunk of basic MIB to XML conversion can be specified and automated fairly easily, but some specific difficul-

> ties remain and, it appears, you also have to consider how certain SNMP aspects will be implemented in the (logical) 'gateway' component. None of the [previous] work appears to have provided for all aspects of SMIv2 and SNMPv2c/v3. MIB2RMDL [(MIB to Resource Model Description Language)] will have to provide such coverage.

In the current context, aspects of these gateways have the potential to address requirements evident in the Semantic Framework depicted in Figure 21.[38] For example, XML-based schemas could be incorporated directly into `Stage I` of this framework as `*ML.xsd` in the Figure. And on a more-generic level, the stylistic representations of MIBs made available by Yoon et al. [76] have the potential to aid in the development of RDFS-based schemas in `Stage II` of the semantic framework. Finally, it is also of interest that one of these gateways has been extended to convert SNMP SMI data into NETCONF data [25]. Unfortunately, this is not a straightforward process either [25]:

> An important difference between SNMP and NETCONF is the separation between state data and configuration data. SNMP does not distinguish between these two kinds of data. Therefore, SMI does not provide any means to specify whether an object shall be considered a configuration object or state data. Thus, a simple straightforward mapping from SMI MIB modules to NETCONF modules is not possible.

Although suggestions are offered, e.g., the use of compiler hints through annotated SMI definitions [25], there remain challenges in effecting these conversions that will need to be addressed as NETCONF encounters increased levels of adoption [15].

```
1 NodeName OBJECT–TYPE
2   SYNTAX "SyntaxType"
3   ACCESS "AccessType"
4   STATUS "StatusType"
5   DESCRIPTION "DescriptionText"
6   REFERENCE "ReferenceType"
```

[38] Incidentally, owing to their architectural complimentarities, it would be a relatively straightforward process to incorporate one of these gateways into the platform provided by Netdisco.

```
7   INDEX "IndexList"
8   DEFVAL "DefaultValue"
9 : : = {parentNodeName nodeNumber}
```

Listing 13: Stylistic description of SNMP SMI's `OBJECT-TYPE` construct from [76].

Motivated by the challenge of scheduling workloads for distributed processing in the context of Grid Computing, Yu & Jin [77] developed an ontology for host resources to abstract away the heterogeneity inherent in these environments. In a spirit analogous to that discussed above, these authors also placed significant emphasis on SNMP SMI datatypes. Using MIBs (presumably) in the manual fashion described previously, i.e., in using Listing 10 to seed development of Listing 11, Yu & Jin developed a host-resources ontology. Noteworthy in this development, was the highly selective extraction of object type data from the MIB to suit their specific purposes. In aiming to balance minimalism with completeness, the authors achieved the outcome they desired. Although the notion of extracting knowledge representations from SNMP MIBs is common to Yu & Jin's work and that presented here (§5.), there are some notable differences. Whereas, for example, Yu & Jin have manually crafted an ontology for a specific requirement, the current effort emphasizes an automated process that can ultimately be applied to any MIB. Furthermore, the motivation for the effort is with respect to knowledge maps for IP networks, as opposed to the challenge of workload scheduling in highly heterogeneous distributed environments.

The notion of extracting RDFS-based knowledge representations from SNMP MIBs is also common to Shen & Yang's work [45] and that presented here (§5.). Shen & Yang provide a representative MIB deconstruction in RDFS and implementation on three separate platforms — notably, none of which is the Redland platform used here in §3.2.. Further research is required to better understand Shen & Yang's approach and thus detail the similarities and differences with the current effort.

From Open Source implementations (e.g., Netdisco, [13]) to commercial products from Independent Software Vendors (e.g., [19, 32]), the discovery and use of network topology is known to be of value. As hinted at above in referring to Yu & Jin's work [77], topology awareness is of even graver interest and consequence in highly heterogeneous distributed environments. Not too surprisingly then in the context of Grid Computing, topology awareness is the subject

of active discourse in the development of standards (e.g., [37]) and implementations (e.g., [7, 26]). In striking contrast to the Grid Computing case, topology awareness is a complexity that is effectively eliminated in the context of Cloud Computing [6]. This suggests that knowledge of IP networks in the latter case of Cloud Computing is likely to be a whole less about topology awareness, and much more about, e.g., auditing against matters of compliance of privacy, security or other concern. Thus the applicability of the approach developed here extends well beyond the realm of mapping the topology of an IP network.

7. Conclusion

Netdisco is an Open Source tool for network management. It is used extensively to manage data pertaining to York University's IP network.[39] That Netdisco is a tool is quite likely an understatement. Argued here (§2.), Netdisco is more of a platform that allows for the discovery of IP networks and then the aggregation of detailed information on the same. Once discovered (see, e.g., Figure 2), Netdisco devices expose their characteristics from a management perspective via SNMP. Specifically, Netdisco employs the `SNMP::Info` Perl module to extract information from MIBs on a per-device basis, and the stores the results in its PostgreSQL database according to a schema. Thus Netdisco produces topologically correct and documented visualizations of the York IP network (e.g., Figure 4). Unfortunately, such visualizations are only practical in certain situations; attempts to visualize an entire campus network (see, e.g., Figure 6), for example, are beyond Netdisco's capabilities as they exist today.

Rather than extend Netdisco directly, the approach taken here was to employ emerging standards and implementations from the Semantic Web (e.g., [1]) in a fashion that is complimentary to Netdisco. Therefore, the starting point for the current investigation was Netdisco's relational database, as it is a repository of data for York's IP network. Using RDF, fields from tables in Netdisco's database were extracted and recast as subject-predicate-object triples (§3.1.). Furthermore, to accelerate this transformation, use was made of the Redland

[39] Netdisco's database schema has been extended to incorporate cable-plant data pertaining to York's legacy telephony service; this service is completely out-of-band from the perspective of the University's IP network. Netdisco user interfaces were also modified to incorporate this modification.

RDF Library [12]. Redland was employed to accept data queried from Netdisco's database and represent it as RDF triples, create an RDF-based model of this data, create a non-volatile record of the same in an RDF store, and allow for serialization of the output in RDF/XML (§3.2.). Instantiated RDF models in Redland are amenable to queries. Thus in §3.3., a SPARQL-based query illustrated the removal of the visualization limitations specific to Netdisco (e.g., Figure 14). And in so doing, an important consequence of the current approach is made explicit: The introduction of a relationship-centric representation also introduces a generalization from topological maps of an IP network to *knowledge maps* of the same. In other words, the resulting visualizations are not restricted to conveying just topological relationships — they are far more generic than that. Furthermore, it is the nature of the query that determines the knowledge of the IP network that will be conveyed.

Even though the relationship-centric representation thus-far developed has intrinsic value, it is placed in the broader context of a Semantic Framework originally developed for use cases in the physical sciences in §4.1. Data passing through this framework experiences a systematic enhancement in its expressivity. Already delivering measures of expressivity common to the second stage of the framework, the relationship-centric representation described above becomes amenable to additional enhancements. Because any XML document can be annotated with XPointer, editorial metadata can be readily incorporated into the framework (§4.2.). As another compelling demonstration of the Web's AAA Slogan, `.rdf` files expressing the relationship-centric representation were annotated; the resulting annotations, themselves `.rdf` files, were validated and visualized.

As the semantic framework of §4.1. (Figure 15) suggests, there is a desire to seek successively more expressive representations. Thus it is not surprising that the very results of the admittedly instance-oriented representations in RDF themselves present an opportunity for framing enhanced expressivity. Motivated by the set-oriented paradigm evidenced in RDF Schema, a vocabulary for IP network knowledge was extracted from the RDF-based representation in §5.1. And in introducing this set-oriented paradigm, RDFS's innate ability for inferencing was shown to be one of the key values of this undertaking. Inferencing was shown, as an example, to allow for very efficient propagation of specific properties (i.e., vendor end-of-life data) to affected hardware (i.e., as-

serted and inferred subclasses of an identified superclass). Bolstered by this significant benefit, approaches for automating the development of RDFS-based schemas lead to a recontextualization of the semantic framework in §5.2. Thus the extraction of RDFS-based schemas from SNMP MIBs is a potential byproduct of application of this recontextualized framework (Figure 21). IDEs such as Protégé were also drawn out as having the potential to refine or even refactor the RDFS-schema resulting from this automated process.

The notion of using MIBs to seed the construction of XML-based representations is a topic that has received attention elsewhere as discussed in §6. From the community whose focus has been SNMP in LAN environments, this attention has been directed at the development of XML-based representations for use in modern architectural contexts (e.g., SOAs, Web Services, etc.). From the perspective of extra-LAN communities, e.g., those having an interest in Grid Computing, RDF, RDFS and even ontological representations have been crafted from SNMP MIBs. Thus one aspect of the current research requiring further investigation, is that of reviewing in additional detail the efforts of others.[40] As noted previously, in the case of SNMP-XML gateways, it may be possible to take advantage of the efforts of others to accelerate closure on a complete and usable framework for SNMP MIB to increasingly expressive XML-based representations.

Acknowledgments

The Semantic Framework (§4.) applied here was developed in collaboration with K. D. Aldridge, J. R. Freemantle and J. I. Lederman; their contributions are graciously acknowledged. Aspects of this research have benefited significantly from Amaya, Protégé and Redland; the individuals and/or communities involved are graciously acknowledged for their significant efforts.

[40] *Added in Proof:* Particularly notable omissions include formal frameworks for network graphs (e.g., [43]), as well as the recent formation of a W3C Working Group whose mission "... is to standardize a language for mapping relational data and relational database schemas into RDF and OWL ..." [66].

References

[1] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[2] B. Natale. Expressing SNMP SMI Datatypes in XML Schema Definition Language. http://tools.ietf.org/html/draft-ietf-opsawg-smi-datatypes-in-xsd-05.

[3] B. Natale. Getting started: Existing MIB to XML and related resource model efforts. http://www.ietf.org/mail-archive/web/mib2rdml/current/msg00006.html.

[4] Max Baker. Netdisco - Easy Network Management for ResNets. *ResNet 2004*, Princeton, New Jersey, USA, June 2004.

[5] T. Berners-Lee. Semantic Web Concepts. http://www.w3.org/2005/Talks/0517-boit-tbl/.

[6] Bhaskar Prasad Rimal and Eunmi Choi and Ian Lumb. A Taxonomy and Survey of Cloud Computing Systems. NCM 2009, IEEE Computer Society, http://www.aicit.org/ncm/, 2009 (accepted).

[7] Mathijs den Burger, Thilo Kielmann, and Henri E. Bal. Topomon: A monitoring tool for grid network topology. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part II*, pages 558–567, London, UK, 2002. Springer-Verlag.

[8] Cisco Systems, Inc. Catalyst 5000 and 5500 Series End of Sale Announcement. http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps679/prod_end-of-life_notice09186a008032d4ae.html.

[9] Cisco Systems, Inc. CISCO-STACK-MIB. http://www.cisco.com/en/???

[10] Cisco Systems, Inc. Troubleshooting the Catalyst 5000 Route Switch Module (RSM) and InterVLAN Routing. http://www.cisco.com/en/US/products/hw/switches/ps679/products-_tech_note09186a00800a7af2.shtml#architecture.

[11] The World Wide Web Consortium. Naming and Addressing: URIs, URLs, ... http://www.w3c.org/Addressing.

[12] Dave Beckett. Redland RDF Libraries. http://librdf.org/.

[13] Eric Miller and Bill Fenner and Oliver Gorwits and Max Baker. Netdisco - Network Management and Discovery. http://www.netdisco.org/.

[14] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Professional Technical Reference. Prentice Hall, 2005.

[15] David French. NETCONF: Ready for Primetime or Work in Progress? http://www.embeddedmind.com/pub/files/ac6a331c.pdf.

[16] Goddard Space Flight Center, National Aeronatics and Space Administration and Information Technology and Systems Center, University of Alabama in Huntsville. Earth Sciences Markup Lanaguage (ESML). http://esml.itsc.uah.edu.

[17] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[18] D. Hazaël-Massieux. Bridging XHTML, XML and RDF with GRDDL. *XTech 2005: XML, the Web and Beyond*, Amsterdam, Netherlands, May 2005.

[19] Hewlett-Packard Company. HP Network Node Manager (NNM): Advanced Edition Software. https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-119%5E1155_4000_100__.

[20] Hewlett-Packard Development Company, LP. Jena - A Semantic Web Framework for Java. http://jena.sourceforge.net/.

[21] Ian Lumb. HTML vs. XHTML: The Five Key Takeaways. http://www.brighthub.com/internet/web-development/articles/33903.aspx.

[22] Jonathan Swartz and Dave Rolsky. Mason HQ. http://www.masonhq.com/.

[23] K. McCloghrie and D. Perkins and J. Schoenwaelder. Structure of Management Information Version 2 (SMIv2). http://tools.ietf.org/html/rfc2578.

[24] T. Klie and F. Straub. Integrating SNMP agents with XML-based management systems. *Communications Magazine, IEEE*, 42(7):76–83, July 2004.

[25] Torsten Klie. Generating Skeleton Code for NETCONF Modules from SMI MIB Module Definitions. In *Proc. IADIS International Conference on WWW/Internet 2005*, volume II, pages 55–59, Lisbon, October 2005.

[26] Sebastien Lacour, Christian Perez, and Thierry Priol. A network topology description model for grid application deployment. In *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 61–68, Washington, DC, USA, 2004. IEEE Computer Society.

[27] Lee Feigenbaum. SPARQL by Example: A Tutorial. http://www.cambridgesemantics.com/2008/09/sparql-by-example/.

[28] I. Lumb and K. D. Aldridge. Grid-enabling the Global Geodynamics Project: The introduction of an XML-based data model. In I. Kotsireas and D. Stacey, editors, *Proceedings of The 19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005*, pages 216–222. The IEEE Computer Society, 2005.

[29] I. Lumb and K. D. Aldridge. Grid-enabling the Global Geodynamics Project: Automatic RDF extraction from the ESML data description and representation via GRDDL. In R. G. Deupree and J. A. Adams, editors, *20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment, HPCS 2006*, pages 184–191. The IEEE Computer Society, 2006.

[30] L. I. Lumb, J. R. Freemantle, J. I. Lederman, and K. D. Aldridge. Annotation modeling with formal ontologies: Implications for informal ontologies. *Comp. & Geosci.*, 35:855–861, 2009.

[31] L. I. Lumb, J. I. Lederman, J. R. Freemantle, and K. D. Aldridge. Semantically enabling the Global Geodynamics Project: Incorporating feature-based annotations via XML Pointer Language (XPointer). In *21st International Symposium on High-Performance Computing, HPCS 2007*, page 21. The IEEE Computer Society, 2007.

[32] Machine-To-Machine Intelligence (m2mi) Corporation. Machine-To-Machine Intelligence (m2mi) Corporation. http://m2mi.com/.

[33] Michael Sintek. OntoViz. http://protegewiki.stanford.edu/index.php/OntoViz.

[34] P. Morville. *Ambient Findability*. O'Reilly & Associates, 2005.

[35] Mozdev.org. Annozilla (Annotea on Mozilla). http://annozilla.mozdev.-org/.

[36] Yoon-Jung Oh, Hong-Taek Ju, Mi-Jung Choi, and James Won-Ki Hong. Interaction Translation Methods for XML/SNMP Gateway. In *DSOM '02: Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, pages 54–65, London, UK, 2002. Springer-Verlag.

[37] Open Grid Forum. Open Grid Forum. http://www.ogf.org/.

[38] David Perkins and Evan McGinnis. *Understanding SNMP MIBs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[39] S. Powers. *Practical RDF*. O'Reilly & Associates, 2003.

[40] Protégé. Protégé. http://protege.stanford.edu/.

[41] Rahul Ramachandran, Sara J. Graves, Helen Conover, and Karen Moe. Earth Science Markup Language (ESML): A solution for scientific data-application interoperability problem. *Comp. & Geosci.*, 30:117–124, 2004.

[42] K. H. Rose, S. Malaika, and R. J. Schloss. Virtual XML: A toolbox and use cases for the XML world view. *IBM Systems Journal*, 45(2):411–424, 2006.

[43] Simon Schenk and Steffen Staab. Networked graphs: A declarative mechanism for SPARQL rules, SPARQL views and RDF data integration on the Web. In *WWW '08: Proceedings of the 17th International Conference on the World Wide Web*, pages 585–594, New York, NY, USA, 2008. ACM.

[44] N. Shadbolt, W. Hall, and T. Berners-Lee. The Semantic Web revisited. *Intelligent Systems*, 21(3):96–101, 2006.

[45] Jun Shen and Yun Yang. RDF-based knowledge models for network management. pages 123–126, March 2003.

[46] M. P. Singh and M. H. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons Ltd., Hoboken, NJ, 2005.

[47] SourceForge.Net. SNMP::Info. http://snmp-info.sourceforge.net/.

[48] The Apache Software Foundation. The Apache HTTP Server Project. http://httpd.apache.org/.

[49] The Dublin Core Metadata Initiative. The Dublin Core Metadata Initiative. http://dublincore.org/.

[50] The Open Grid Forum. Data Format Description Language Working Group. http://forge.gridforum.org/projects/dfdl-wg.

[51] The Perl Foundation. The Perl Foundation. http://www.perlfoundation.org/.

[52] The University of California at Berkeley. PostgreSQL. http://www.postgresql.org/.

[53] Tim Bunce and Ilya Sterin. Perl DBI. http://dbi.perl.org/.

[54] Inc. Wikimedia Foundation. Grace hopper. http://en.wikiquote.org/wiki/Grace_Hopper.

[55] Wikipedia. Cisco Discovery Protocol. http://en.wikipedia.org/wiki/Cisco_Discovery_Protocol.

[56] Wikipedia. Simple Network Management Protocol. http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol.

[57] Wikipedia. SQL. http://en.wikipedia.org/wiki/SQL.

[58] World Wide Web Consortium. Amaya Home Page. http://www.w3.org/-Amaya/.

[59] World Wide Web Consortium. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). http://www.w3.org/2004/01/rdxh/spec/.

[60] World Wide Web Consortium. GRDDL Primer, Editor's Draft 27 September 2006. http://research.talis.com/2006/grddl-wg/primer.html.

[61] World Wide Web Consortium. Notation3 (N3) A readable RDF syntax. http://www.w3.org/DesignIssues/Notation3.

[62] World Wide Web Consortium. OWL 2 Web Ontology Language Primer. http://www.w3.org/TR/owl2-primer/.

[63] World Wide Web Consortium. OWL Web Ontology Language Overview. http://www.w3c.org/TR/owl-features.

[64] World Wide Web Consortium. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/.

[65] World Wide Web Consortium. Parsing OWL in RDF/XML, W3C Working Group Note 21 January 2004. http://www.w3.org/TR/owl-parsing/.

[66] World Wide Web Consortium. RDB2RDF Working Group Charter. http://www.w3.org/2009/08/rdb2rdf-charter.html.

[67] World Wide Web Consortium. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/.

[68] World Wide Web Consortium. Resource Description Framework (RDF). http://www.w3.org/RDF/.

[69] World Wide Web Consortium. SPARQL Query Language for RDF. http://-www.w3c.org/TR/rdf-sparql-query.

[70] World Wide Web Consortium. W3C Document Object Model. http://www.w3.org/DOM/.

[71] World Wide Web Consortium. W3C RDF Validation Service. http://www.w3.org/RDF/Validator/.

[72] World Wide Web Consortium. W3C Recommendation (REC). http://-www.w3.org/2005/10/Process-20051014/tr.html#RecsW3C.

[73] World Wide Web Consortium. Web Services Architecture. http://www.w3.org/TR/ws-arch/.

[74] World Wide Web Consortium. XPointer xpointer() Scheme. http://www.w3.org/TR/xptr-xpointer/.

[75] World Wide Web Consortium. XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt.

[76] Jeong-Hyuk Yoon, Hong-Taek Ju, and James W. Hong. Development of SNMP-XML translator and gateway for XML-based integrated network management. *Int. J. Netw. Manag.*, 13(4):259–276, 2003.

[77] Yijiao Yu and Hai Jin. An ontology-based host resources monitoring approach in grid environment. In *Advances in Web-Age Information Management*, pages 834–839, London, UK, 2005. Springer Berlin / Heidelberg.