Declarative Construction of Knowledge Graphs from NETCONF Data Sources

Ignacio Domínguez Martínez-Casanueva [a,b,*], Luis Bellido [a] and Diego López [b]
[a] *Dpto. de Ingeniería de Sistemas Telemáticos, ETSI Telecomunicación, Universidad Politécnica de Madrid, Spain*
E-mails: i.dominguezm@alumnos.upm.es, luis.bellido@upm.es
[b] *GCTIO, Telefónica Innovación Digital, Spain*
E-mail: diego.r.lopez@telefonica.com

Abstract. The knowledge graph paradigm is drawing attention in the network industry as a technology for integrating heterogenous data silos such as model-driven telemetry based on the YANG language. In this sense, declarative mapping languages have emerged as scalable and flexible solutions to construct knowledge graphs. A prominent mapping language is the Resource Mapping Language (RML), which enables the integration of heterogeneous data sources by reusing ontologies that describe access to them. However, when it comes to the network domain, there is a lack of ontologies that describe access to YANG data exposed by network devices. This paper introduces the YANG Server Ontology for describing YANG servers and the interactions with them using network protocols such as NETCONF. Additionally, guidelines for reusing the ontology in RML mappings are provided and validated in a use case by extending a reference RML engine.

Keywords: Knowledge Graphs, Mapping Languages, Ontology Description, Network Management, YANG

1. Introduction

The creation of knowledge graphs based on declarative mapping languages, such as the Relational to RDF Mapping Language (R2RML) or the Resource Mapping Language (RML), is gaining traction [1]. Until now, these languages have focused on general-purpose data sources such as relational databases, remote files, or message brokers. However, in the scope of network management, in addition to these general purpose data sources, there are industry-specific data sources widely used. One of the latest trends is model-driven telemetry (MDT), which is based on the YANG data modeling language [2]. To access the data available on YANG-capable network devices, a variety of network management protocols can be used, such as NETCONF [3], RESTCONF [4], or gNMI [5]. In this sense, to enable the ingestion and integration of YANG data into a knowledge graph, declarative mapping engines must implement these protocols and cope with the intricacies of YANG [6].

The complexity of network infrastructures continues to grow; thus, the implementation of a network telemetry framework that can provide insights on the status of the network is becoming a relevant topic for the industry [7]. The ingestion, correlation, and consumption of large amounts of data obtained at multiple planes of the network can enable new use cases in network management such as root cause analysis, anomaly detection, or network digital twins [8]. To this end, a telemetry framework must deal with a diversity of network telemetry techniques that involve different data models, formats, and protocols. In this regard, the knowledge graph has been identified as a perfect candidate due to its capabilities for semantic data integration.

[*]Corresponding author. E-mail: i.dominguezm@alumnos.upm.es.

Among these network telemetry techniques, the success of YANG has made supporting model-driven telemetry one of the priorities of operators. However, since the advent of YANG, the network industry has experienced a rapid evolution, with numerous network vendors implementing the YANG language and network management protocols on their devices and controllers. This evolution led to YANG data models created by vendors, standards-developing organizations, and open-source communities. The result is a plethora of independent YANG data models that hinder data correlation and follow different schemas but often refer to the same concepts. Adding to this divergence, YANG is a logical data modeling language that decouples the data model from the encoding format and the access protocol. For instance, the NETCONF protocol encodes YANG data in XML format and relies on SSH as the transport protocol, whereas the RESTCONF protocol encodes in JSON format and uses the HTTP protocol. In summary, the construction of knowledge graphs from YANG-based data sources becomes a challenge.

This work tackles the declarative integration of YANG data into knowledge graphs by means of the YANG Server Ontology, which enables the description of YANG servers and network management operations to retrieve YANG data from those servers. This first version of the ontology focuses on describing the NETCONF protocol, though it has been designed to enable future extensions for other protocols. Additionally, this work provides guidelines for referencing the YANG Server Ontology in declarative mappings based on the RML language, thus allowing the description of YANG servers as logical sources in the construction of knowledge graphs.

The remainder of this paper is structured as follows. Section 2 provides an overview of declarative mapping languages for the construction of knowledge graphs and their application in the scope of network management. Section 3 introduces the YANG Server Ontology, describing the methodology followed for its development. Section 4 describes the alignment of the YANG Server Ontology with the RML language to declare the YANG servers as logical sources in the creation of knowledge graphs. Section 5 presents a practical use case with a prototype that demonstrates the applicability of the YANG Server Ontology combined with RML to create a knowledge graph. Section 6 draws conclusions and identifies future work lines.

2. Related Work

Knowledge graphs have been broadly explored as data integration enablers in different areas of telecommunication network management, ranging from mobile to SDN-based networks [9]. The applications of knowledge graphs in these areas have varied, from supporting network operations to delivering service assurance, but none of them have followed declarative mechanisms to construct their knowledge graphs. The NORIA project proposes a declarative construction of knowledge graphs for anomaly detection based on two data pipelines to integrate static data from remote files and events consumed from Apache Kafka [10]. However, NORIA does not address the access to data from YANG-based data sources. In this sense, the integration of YANG data into knowledge graphs has just started to draw the attention of the network industry with recent novel contributions. In [11], the authors discuss the motivations and challenges around the creation and use of knowledge graphs for network operations. Furthermore, in the CANDIL initiative [12], the authors propose the declarative transformation of the YANG data into RDF using RML; however, as in the case of NORIA, the YANG data are ingested through Apache Kafka instead of directly from the YANG data sources using network management protocols such as NETCONF.

The evolution of knowledge graphs and the growing general interest in them has sparked the development of tools and languages that can facilitate their construction. In this regard, declarative mapping languages provide a scalable and flexible approach to transform heterogeneous data into RDF according to a target ontology. More than a decade ago, the R2RML language became a W3C recommendation [13], which defined a standard language to map data from relational databases into RDF. Over the years, the wide adoption of R2RML in real use cases showed limitations that extensions and new mapping languages aimed to address [14]. Among these initiatives, RML was conceived as an extension of R2RML to overcome the limitations of this language and add support for transforming semi-structured data sources like JSON, XML, or CSV into RDF. Following on this effort, the W3C Community Group on Knowledge Graph Construction (KGC) [15] was established, and over the last five years, its large community has been working on standardizing a modular ontology for the RML language [16].

Among the modules that comprise the RML ontology, the RML-IO module[1] focuses on formal representations to describe access to data sources and destinations in the creation of knowledge graphs. RML-IO advocates for the reuse of existing ontologies that describe access to data sources and targets; thus, RML-IO does not limit itself to specific data sources or targets. In this sense, the community has already identified several relevant ontologies that can be leveraged, such as D2RQ [17] to access relational databases, DCAT [18] for remote files, and the Web of Things [19] for Web APIs and data streams such as MQTT or Apache Kafka. These are some examples of ontologies that can be used to describe access to general-purpose data sources in RML mappings, however, when it comes to the scope of model-driven network telemetry, we could not find ontologies to describe access to YANG data sources like NETCONF.

3. YANG Server Ontology

The YANG Server Ontology has been developed by following the guidelines defined in the Linked Open Terms (LOT) methodology [20]. LOT is a mature and lightweight methodology for the development of ontologies that embraces the best practices of agile software development. The methodology iterates over a workflow composed of four activities: 1) Ontology requirements specification; 2) Ontology implementation; 3) Ontology publication; 4) Ontology maintenance. The following subsections describe how the different activities have been conducted during the development of the YANG Server Ontology.

3.1. Requirements Specification

The goal of the YANG Server Ontology is to enable the description of YANG servers and management operations such as queries or subscriptions. The ontology aims to capture common aspects across the different existing network management protocols, such as NETCONF or RESTCONF, while allowing future extensions of the ontology to provide further details for each protocol. In this sense, this work already provides an extension of the ontology with details on the NETCONF protocol for two reasons: i) to serve as a reference for other protocol extensions; and ii) to demonstrate the applicability of the ontology in a practical use case.

The ontological requirements were manually extracted based on interviews that involved several members of the Telefonica network operations team and representatives of two additional network service providers. These interviews were aimed at identifying and describing use cases by experts with long experience in YANG-based network management. In addition, the results of these interviews were complemented by a thorough analysis of a set of specifications that contain standard data models and documentation that have been agreed by experts in the network industry [3, 6, 21, 22]. The results of the analysis were shared and discussed with representatives of multiple network vendors with experience in operations and standardization within the IETF. This step helped to specify the subdomains of the ontology and identify their respective core concepts as detailed in the next sub-section.

The specification of the requirement was performed over several sprints, each tackling a new subdomain. For example, a sprint was dedicated to the analysis of the concepts of a YANG server and a YANG datastore, another sprint addressed query and subscription operations, while another sprint focused on technical details of the NETCONF protocol. The requirements were captured in the form of natural language statements and stored in a CSV file[2] to facilitate their processing and visualization. The requirements are specified as facts rather than competency questions, as they were easier to validate by network experts without prior knowledge of ontologies and the SPARQL query language.

3.2. Implementation

Based on the ontological requirements specified in the previous step, this activity has implemented the ontology using a formal language. For conceptualization, the ontology has been represented in a diagram following the

[1]https://w3id.org/rml/io/spec
[2]https://github.com/candil-data-fabric/yang-server-ontology/blob/main/requirements/requirements.csv

Chowlk notation [23], as shown in Fig. 1. The ontology has been implemented in the OWL 2 language using the Protégé tool. Finally, the overall quality of the ontology has been evaluated and improved using the OOPS! tool [24] to identify common pitfalls and the FOOPS! tool [25] to align with the FAIR principles. The reports generated by these tools, along with the SPARQL queries that validate the competency questions, were uploaded to the repository [3]. In the following subsections, the YANG Server Ontology is explained in detail along with a series of RDF examples.

3.2.1. YANG Server Core

The core of the ontology revolves around the concept of a YANG server (`ys:YangServer`), which represents a YANG-capable network element, such as a device or a controller. This class includes connection and authentication details to access a YANG server. The connection endpoint of the YANG server (`ys:socketAddress`) is represented by reusing the concept `observable:SocketAddress` from the UCO Ontology [26], which describes the endpoint as a socket address (`observable:addressValue`). Similarly, the account used to authenticate with the server (`ys:ServerAccount`) is represented by reusing the terms `foaf:OnlineAccount` and `observable:UserAccount` from the FOAF [27] and UCO ontologies, respectively. Both ontologies define the concept of username (`ys:username`) with `foaf:accountName` and `observable:userName`. Furthermore, the term `observable:AccountAuthenticationFacet` from the UCO Ontology is also used to represent the password (`observable:password`) of the account, for basic authentication.

On the other hand, the core concept of the YANG datastore (`ys:Datastore`) is also defined in the ontology. A datastore represents the conceptual place to store and access YANG data. According to the Network Management Datastore Architecture (NMDA) [21], a server might have to run multiple datastores simultaneously for different purposes. For example, the "running" datastore contains the current configuration of the device, whereas the "operational" datastore holds the complete operational status of the device (i.e., the applied configuration plus the system state). To represent the different types of datastore that a server could run, a hierarchy of subclasses of `ys:Datastore` has been defined based on the NMDA specification. In this sense, the running and the operational datastores are represented by `ys:RunningDatastore` and `ys:OperationalDatastore` classes respectively.

The ontology follows a design choice that models the YANG datastore as a separate class. This choice facilitates the representation of YANG operations, such as queries, by indicating the target datastore without having to indicate the server as well. Moreover, this approach improves inventory management because, in addition to the standardized YANG datastores, devices compatible with the NMDA architecture can run custom datastores defined by vendors. Lastly, in the use case of a YANG catalog, the knowledge graph would capture the YANG modules implemented by the datastores running on a given device. More details on this last example are provided later in Section 5.

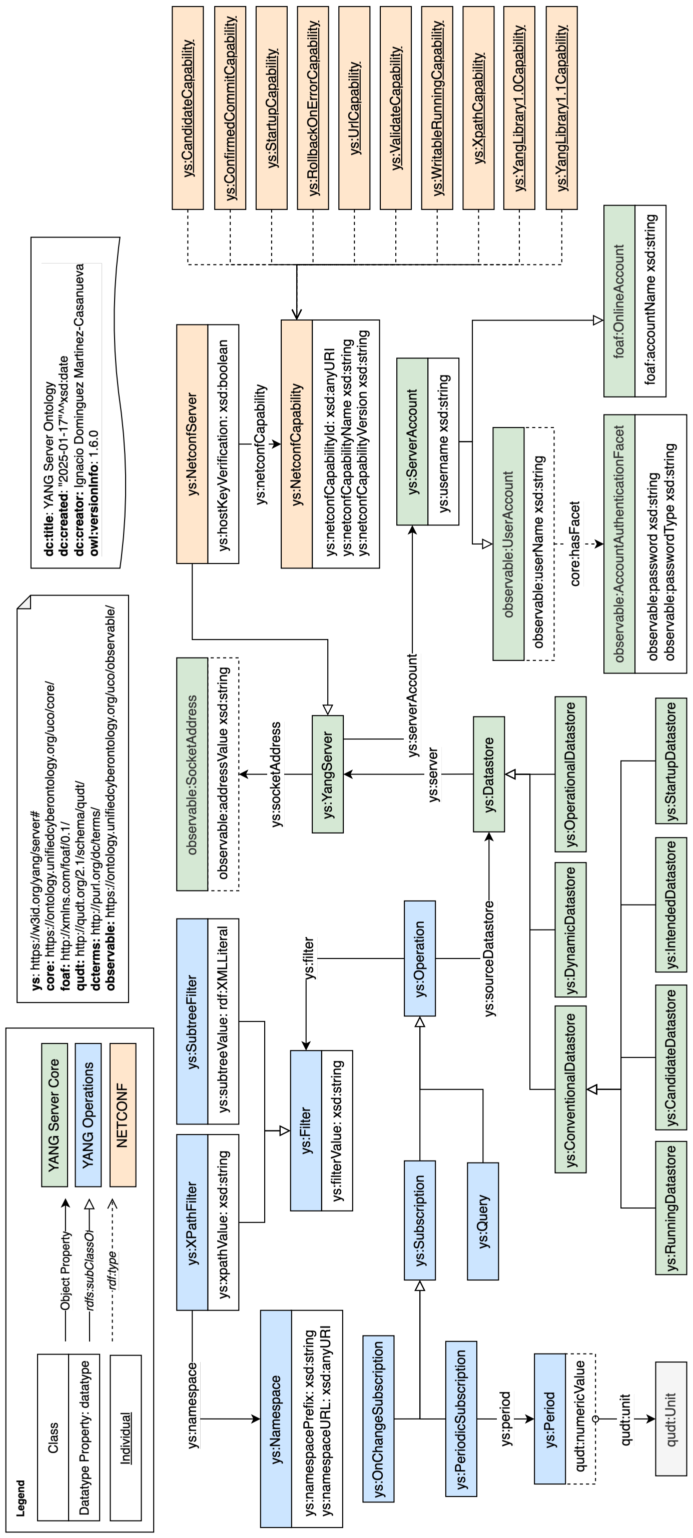
3.2.2. NETCONF

The ontology captures the concept of the YANG server as a generic class, so extensions can add support for network management protocols such as NETCONF or RESTCONF through subclasses derived from `ys:YangServer`. This version of the ontology includes as a reference an extension that includes additional concepts related to the NETCONF network management protocol. First, `ys:NetconfServer` is defined as a subclass of `ys:YangServer` to denote a YANG server that implements the NETCONF protocol. This class includes a flag to enable the verification of host keys (`ys:hostKeyVerification`) in the configuration of the SSH session with the server. Additionally, the ontology extension allows us to describe the NETCONF capabilities (`ys:NetconfCapability`) supported by the server. In this regard, a set of standard capabilities have already been incorporated into the ontology such as `ys:XpathCapability`, which indicates that the server supports XPath filtering (i.e., `ys:XPathFilter`).

3.2.3. YANG Operations

Network management operations to retrieve data from a YANG server are covered by the ontology using the class `ys:Operation`. More specifically, queries (`ys:Query`) and subscriptions to notifications (`ys:Subscription`) have been defined as types of operations. In a query, a single request is sent to the server to retrieve the latest data. On the other hand, in a subscription, the client creates a session with the server and waits to receive notifications pushed by the server when a specific type of event occurs.

[3]https://github.com/candil-data-fabric/yang-server-ontology/tree/main/evaluation


ys:CandidateCapability
ys:ConfirmedCommitCapability
ys:StartupCapability
ys:RollbackOnErrorCapability
ys:UrlCapability
ys:ValidateCapability
ys:WritableRunningCapability
ys:XpathCapability
ys:YangLibrary1.0Capability
ys:YangLibrary1.1Capability
dc:title: YANG Server Ontology
dc:created: "2025-01-17"^^xsd:date
dc:creator: Ignacio Dominguez Martinez-Casanueva
owl:versionInfo: 1.6.0
ys:NetconfServer
ys:hostKeyVerification: xsd:boolean
ys:netconfCapability
ys:NetconfCapability
ys:netconfCapabilityId: xsd:anyURI
ys:netconfCapabilityName xsd:string
ys:netconfCapabilityVersion xsd:string
ys:ServerAccount
ys:username xsd:string
foaf:OnlineAccount
foaf:accountName xsd:string
observable:UserAccount
observable:userName xsd:string
core:hasFacet
observable:AccountAuthenticationFacet
observable:password xsd:string
observable:passwordType xsd:string
ys: https://w3id.org/yang/server#
core: https://ontology.unifiedcyberontology.org/uco/core/
foaf: http://xmlns.com/foaf/0.1/
qudt: http://qudt.org/2.1/schema/qudt/
dcterms: http://purl.org/dc/terms/
observable: https://ontology.unifiedcyberontology.org/uco/observable/
observable:SocketAddress
observable:addressValue xsd:string
ys:socketAddress
ys:YangServer
ys:serverAccount
ys:server
ys:Datastore
ys:OperationalDatastore
ys:StartupDatastore
ys:DynamicDatastore
ys:IntendedDatastore
ys:ConventionalDatastore
ys:CandidateDatastore
ys:RunningDatastore
ys:sourceDatastore
ys:Operation
ys:filter
ys:SubtreeFilter
ys:subtreeValue: rdf:XMLLiteral
ys:Filter
ys:filterValue: xsd:string
ys:XPathFilter
ys:xpathValue: xsd:string
ys:Subscription
ys:Query
Legend
Class
Datatype Property: datatype
Individual
Object Property
rdfs:subClassOf
rdf:type
YANG Server Core
YANG Operations
NETCONF
ys:namespace
ys:Namespace
ys:namespacePrefix: xsd:string
ys:namespaceURL: xsd:anyURI
ys:OnChangeSubscription
ys:PeriodicSubscription
ys:period
ys:Period
qudt:numericValue
qudt:unit
qudt:Unit
Fig. 1. Conceptual representation of YANG Server Ontology (Represented using the Chowlk visual notation [23]).

The creation of a query requires indicating the datastore to obtain the data from, as well as a filter (`ys:Filter`) for selecting a subset of data from the server. Two possible types of filters can be used in a query: XPath (`ys:XPathFilter`) and XML subtree (`ys:SubtreeFilter`). The XPath filter describes the selection of data using an XPath expression along with a map of XML namespaces and prefixes. The subtree filter offers a fine-grained mechanism for selecting the data to be filtered and returned. Please, note that support for these types of filters will be determined by the capabilities of the server and the protocols used (e.g., subtree filters are only supported by NETCONF).

Regarding YANG subscriptions, a datastore and a filter must be specified, but also additional aspects that will depend on the type of subscription. A subscription may trigger notifications on updates to the data selected by the filter (`ys:OnChangeSubscription`). Alternatively, a subscription can trigger notifications periodically based on the specified time interval (`ys:PeriodicSubscription`). In this case, the QUDT Ontology [28] has been leveraged to represent the interval used by the subscription.

3.3. Publication

This activity focuses on documenting and making the ontology publicly available. The documentation has been generated using the WIDOCO tool [29] and published online at http://w3id.org/yang/server/. Ontology artifacts, such as requirement specification, diagram, code, or documentation, are available in a GitHub repository[4]. Furthermore, the ontology has been registered in the Linked Open Vocabulary (LOV) service [30] to improve its discoverability.

3.4. Maintenance

The last activity of the methodology tackles the incorporation of new ontological requirements as well as the identification and fixing of any bugs found in the ontology. In this regard, given that the ontology artifacts are available and tracked in a GitHub repository, this work proposes leveraging GitHub functionalities like issue tracking. Similarly, ontology releases are managed as releases with tags registered on GitHub.

4. RML and YANG Server Ontology Alignment

In addition to the development of the YANG Server Ontology, this paper proposes an alignment with the RML Ontology to declare the construction of knowledge graphs from YANG data sources by using network management protocols. In this sense, the `ys:YangServer` concept cannot be directly treated as the `rml:Source` of the RML mapping, which would seem the natural solution. As described in Section 3, YANG datastores act as separate databases running on the same server, therefore, YANG operations target a specific YANG datastore of the server. Moreover, in addition to indicating the source datastore, YANG operations for data retrieval, such as queries and subscriptions, define a filter in the form of an XPath expression or an XML subtree to filter out data at the source. From the perspective of RML mappings, filtering out data at the source before receiving and iterating the data in the RML engine brings multiple benefits. By filtering data at the server, the data size can be greatly reduced, thus improving network bandwidth usage and time execution in the RML engine. This is a common practice followed for other types of data sources that implement query APIs, such as relational databases, where the D2RQ Ontology is used to select a subset of a whole table at the source before iterating the data in an RML engine. For these reasons, YANG operations, which in turn indicate the source datastore (`ys:Datastore`) and the filter used in the operation (`ys:Filter`), are identified as the sources (`rml:Source`) in the RML mappings.

Fig. 2 shows the proposed alignment between the RML Ontology and the YANG Server Ontology. The alignment defines the superclass `ys:Operation`, which represents a data retrieval operation, as the range of `rml:source`. This design choice allows for reusing previously defined operations or filters, to integrate the retrieved YANG data into

[4]https://github.com/candil-data-fabric/yang-server-ontology

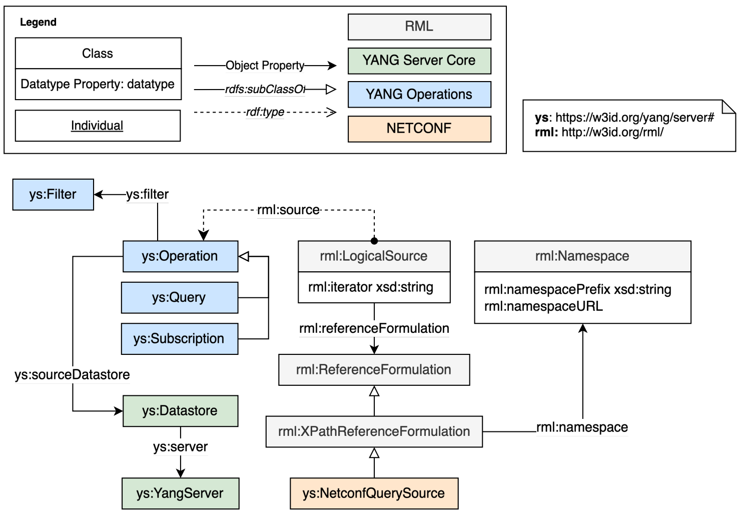


Fig. 2. Alignment between the RML Ontology and the YANG Server Ontology.

a knowledge graph by using RML. Additionally, iterations over the YANG data are decoupled from the operation that retrieves the data, therefore, the same operation can be leveraged by multiple mapping definitions.

However, the type of network management protocol supported by the YANG server will determine the final representation of the logical source in the RML mappings. For example, the NETCONF protocol encodes YANG data in XML format; thus, the RML mappings need to be adjusted to iterate over XML data. Additionally, the RML engine must determine the data collection mechanisms to be used for the protocol. In the example of NETCONF, the YANG Server Ontology defines `ys:NetconfQuerySource` as a new type of reference formulation to indicate to the RML engine that a NETCONF query operation must be executed. This new type has been defined as a subclass of `rml:XPathReferenceFormulation` because the iterations over the retrieved XML data require a definition of XML namespaces. Future implementations such as NETCONF subscriptions or other network management protocols can be supported by defining specific reference formulations.

5. Use Case: Evolution of the YANG Catalog

YANG Catalog[5] is a web service that provides a tool to search existing YANG modules. The YANG Catalog service provides additional metadata such as the name or revision date of a YANG module, the dependencies with other modules, as well as the network devices that implement these modules. Currently, this service periodically scrapes public repositories over the Internet, collecting and storing YANG metadata, and exposing them through a REST API.

The present use case proposes building a knowledge graph that aims at evolving the current YANG Catalog service in two aspects: i) by storing and representing the YANG module metadata in a graph structure, the traversal of the different types of dependencies among modules is optimized; and ii) by facilitating the integration of YANG module metadata with other data silos thanks to a semantic layer. The second aspect proves the benefits of the knowledge graph as a technology to break data silos. For example, network operators can gain insight into the YANG modules implemented by devices deployed in virtualized networks by integrating the metadata of the modules with custom network topology descriptors used by network virtualization environments, such as Containerlab [31] or Kubernetes Network Emulation (KNE) [32]. Similarly, metadata could also be linked in the knowledge graph with network concepts collected from formal vocabularies, enabling semantic-based searches such as: *Does a given*

[5]https://www.yangcatalog.org

device implement any YANG module to manage the interfaces? If so, which module and respective dependencies should I use?.

To create a knowledge graph that supports the YANG Catalog, the proposed approach is based on collecting data from the YANG Library from devices running in the network. YANG Library [33] is a YANG module that provides information about the YANG modules implemented and imported by a YANG server. In this regard, the RML engine responsible for creating the knowledge graph interacts with the network device via a protocol such as NETCONF to ingest the YANG Library data and, based on the declared mappings to the target ontology, integrates the data in the knowledge graph.

The use case begins with a network that runs a NETCONF capable device implemented with netopeer2[6]. The network operator's knowledge graph registers the connection and authentication details to access this device, along with the YANG datastores and NETCONF capabilities supported by the device. This information is represented in the knowledge graph using the YANG Server Ontology as shown in Listing 1.

Listing 1: Details of the NETCONF server.

```
@prefix yl: <https://w3id.org/yang/library#> .
@prefix ys: <https://w3id.org/yang/server#> .
@prefix rml: <http://w3id.org/rml/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix core: <https://ontology.unifiedcyberontology.org/uco/core/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix observable: <https://ontology.unifiedcyberontology.org/uco/observable/> .
@base <https://netconf-rml-demo.org/> .

# Connection details to NETCONF server
<netconf-server-1> a ys:NetconfServer ;
    ys:socketAddress <netconf-server-1/socket-address> ;
    ys:serverAccount <netconf-server-1/account> ;
    ys:hostKeyVerification "false" ;
    ys:capability ys:XpathCapability ,
                  ys:YangLibrary1.0 .

<netconf-server-1/datastores/operational> a ys:OperationalDatastore ;
    ys:server <netconf-server-1> .

<netconf-server-1/datastores/running> a ys:RunningDatastore ;
    ys:server <netconf-server-1> .

<netconf-server-1/socket-address> a observable:SocketAddress ;
    observable:addressValue "localhost:830" .

<netconf-server-1/account> a ys:ServerAccount ;
    ys:username "netconf" ;
    core:hasFacet <netconf-server-1/account/authentication> .

<netconf-server-1/account/authentication> a observable:AccountAuthenticationFacet ;
    observable:password "netconf" ;
```

Analyzing this graph, the operator finds that the NETCONF server supports XPath filtering to retrieve data, in addition to XML Subtree filtering, which is supported by default in NETCONF. Additionally, the server implements the YANG Library version 1.0, which indicates that the server does not support the NMDA architecture. Based on these insights, the operator defines an XPath filter expression that will obtain all the data from the YANG Library of the device (see Listing 2).

Listing 2: XPath filter to retrieve YANG Library 1.0 data.

```

<filters/xpath/yang-library> a ys:XPathFilter ;
    ys:xpathValue "/yanglib:modules-state";
```

[6]https://github.com/CESNET/netopeer2

```
  ys:namespace [ a ys:Namespace ;
    ys:namespacePrefix "yanglib" ;
    ys:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-yang-library" ;
  ];
```

Now, the operator defines a NETCONF query that, using the previous XPath filter, retrieves the YANG Library data from the operational datastore of the NETCONF server. In turn, this query is defined as the source of the RML mapping that will transform the YANG data into RDF based on a target ontology. In this case, the `ys:NetconfQuerySource` reference formulation is selected to indicate the RML engine to send a NETCONF query operation. A fragment of the RML mapping showing the logical source is shown in Listing 3.

Listing 3: NETCONF query as logical source in the RML mapping.

```
<#TriplesMap> a rml:TriplesMap;
  rml:logicalSource [ a rml:LogicalSource;
    rml:source [ a ys:Query, rml:Source ;
      ys:sourceDatastore <netconf-server-1/datastores/operational> ;
      ys:filter <filters/xpath/yang-library>
    ];
    rml:referenceFormulation [ a ys:NetconfQuerySource ;
      rml:namespace [ a rml:Namespace ;
        rml:namespacePrefix "yanglib" ;
        rml:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-yang-library" ;
      ];
    ];
    rml:iterator "/yanglib:modules-state/yanglib:module";
```

To validate this solution, the prototype depicted in Fig. 3 was developed. The selected RML engine was BURP [34], which is currently developed under the umbrella of the KGC Community Group as the reference implementation compliant with the latest RML specification. For this prototype, BURP has been extended[7] with three new features: i) support for connecting to NETCONF sources as logical sources in RML based on the YANG Server Ontology; ii) implementation of YANG query operations, either by using XPath or subtree filters, to access the YANG data from the NETCONF server; and iii) capability to process the newly defined `ys:NetconfQuerySource` to iterate over the XML data retrieved with a NETCONF query using an XPath expression along with a map of XML namespaces and prefixes.

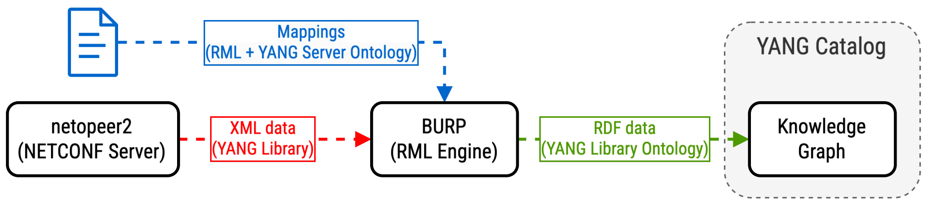


Fig. 3. Prototype overview. Based on the mappings, BURP obtains the YANG Library data from the NETCONF server using the NETCONF protocol. The RDF data generated by BURP are stored in a knowledge graph, which acts as the database of the YANG Catalog service.

For this use case, RML mappings have been developed to integrate the YANG Library 1.0 data into the knowledge graph referencing the YANG Library Ontology[8]. However, it is important to highlight that these mappings are not complete due to current limitations in BURP concerning conditional transformations and iterating nested data. In this sense, the KGC community has identified such limitations and is working on specifications for the RML-FNML module [9] and the RML-LV module [10]. Examples of the input YANG Library data, the mappings, and the generated RDF data have also been uploaded to the YANG Library repository [11].

[7]https://github.com/kg-construct/BURP/pull/11
[8]http://w3id.org/yang/library/
[9]https://kg-construct.github.io/rml-fnml/spec/docs/
[10]https://kg-construct.github.io/rml-fnml/spec/docs/
[11]https://github.com/candil-data-fabric/yang-library-ontology/tree/main/knowledge-graph

6. Conclusions and Future Work

This paper has introduced the YANG Server Ontology, which allows for representing YANG servers and network management operations. The ontology has been developed following a well-known knowledge engineering methodology and has been designed to enable the future extension of the ontology with further details specific to network protocols. This first version has demonstrated this approach by extending the ontology to support YANG servers that implement the NETCONF protocol. Furthermore, this work has provided recommendations and examples that illustrate how the YANG Server Ontology can be aligned with the RML Ontology to declare YANG operations as logical sources in the creation of knowledge graphs. This has been validated with a prototype that also proved the applicability of the YANG Server Ontology and knowledge graphs in a use case within the scope of network management.

Nevertheless, multiple challenges were identified during this work. The LOT methodology provides detailed descriptions for executing the activities defined in the workflow, although some of the steps, such as the collection of requirements, were not easy to follow by network engineers with no prior experience in semantic modeling. Additionally, this work has addressed the YANG ecosystem, which is very specific to the IETF community, and given that knowledge graphs and RML have just started to gain traction, only a few prior works of YANG related with knowledge graphs could be found.

Future extensions of this work could tackle the implementation of YANG subscriptions in existing RML engines. YANG servers that support subscriptions would enable the construction of knowledge graphs on a streaming basis, thus unlocking real-time use cases such as fault detection or network performance. The proposed YANG Server Ontology has addressed the concept of a YANG subscription, but its combination with the RML language has not been tackled yet. Compared to a batch approach in which the RML engine pulls YANG data from a YANG server, a YANG subscription entails a streaming-based approach in which the RML engine creates a session with the YANG server through which YANG notifications would be received (e.g., by means of YANG Push [35]). In this sense, RML engines dedicated to stream processing, such as Streaming MASSIF [36] and RMLStreamer [37] or frameworks like Chimera [38], have been identified as candidates that can be extended to support YANG subscriptions.

Lastly, the YANG Server Ontology can be extended with new concepts to incorporate YANG data sources that implement other network management protocols like RESTCONF or gNMI. The latter, in particular, builds on the gRPC protocol[12] and the Protocol Buffers[13] encoding format, which have not yet been explored by the KGC Community Group. To this end, the combination of the YANG Server Ontology with RML will be adjusted accordingly, and support for these new network management protocols will be added to the RML engines.

Acknowledgements

This work has been partially supported by the European Union's Horizon Europe projects aerOS (Grant Agreement No. 101069732) and ROBUST-6G (Grant Agreement No. 101139068).

References

[1] J. Arenas-Guerrero, M. Scrocca, A. Iglesias-Molina, J. Toledo, L. Pozo-Gilo, D. Doña, O. Corcho and D. Chaves-Fraga, Knowledge Graph Construction with R2RML and RML: An ETL System-Based Overview, in: *CEUR Workshop Proceedings*, Vol. 2873, CEUR-WS, 2021. ISSN 1613-0073.

[2] B. Claise, J. Clarke and J. Lindblad, *Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI*, Pearson Education, 2019. ISBN 978-0-13-518061-7.

[3] R. Enns, M. Björklund, A. Bierman and J. Schönwälder, Network Configuration Protocol (NETCONF), *Request for Comments*, RFC Editor, 2011. doi:10.17487/RFC6241.

[12]https://grpc.io
[13]https://protobuf.dev

[4] A. Bierman, M. Björklund and K. Watsen, RESTCONF Protocol, *Request for Comments*, RFC Editor, 2017. doi:10.17487/RFC8040.

[5] OpenConfig, gNMI - gRPC Network Management Interface, 2023.

[6] M. Björklund, The YANG 1.1 Data Modeling Language, *Request for Comments*, RFC Editor, 2016. doi:10.17487/RFC7950.

[7] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia and A. Wang, Network Telemetry Framework, Request for Comments, RFC 9232, Internet Engineering Task Force, 2022. doi:10.17487/RFC9232.

[8] P. Almasan, M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, D. Perino, D. López, A.A.P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio and P. Barlet-Ros, Network Digital Twin: Context, Enabling Technologies, and Opportunities, *IEEE Communications Magazine* **60**(11) (2022), 22–27. doi:10.1109/MCOM.001.2200012.

[9] F. Javier Zorzano Mier and C.Á. Iglesias, Applications of Knowledge Graphs in Telecommunication Systems Management, *IEEE Internet Computing* **27**(3) (2023), 29–34. doi:10.1109/MIC.2023.3253305.

[10] L. Tailhardat, Y. Chabot and R. Troncy, Designing NORIA: A Knowledge Graph-Based Platform for Anomaly Detection and Incident Management in ICT Systems, in: *KGC 2023, 4th International Workshop on Knowledge Graph Construction, Co-Located with ESWC 2023, 28 May 2023, Hersonissos, Greece*, CEUR, ed., Hersonissos, 2023.

[11] M. Mackey, B. Claise, T. Graf, H. Keller, D. Voyer and P. Lucente, Knowledge Graph Framework for Network Operations, Internet-Draft, draft-mackey-nmop-kg-for-netops-01, Internet Engineering Task Force / Internet Engineering Task Force, 2024.

[12] I.D. Martinez-Casanueva, L. Bellido, D. González-Sánchez and D. Lopez, CANDIL: A Federated Data Fabric for Network Analytics, *Future Generation Computer Systems* **158** (2024), 98–109. doi:10.1016/j.future.2024.04.013.

[13] S. Das, S. Sundara and R. Cyganiak, R2RML: RDB to RDF Mapping Language, 2010.

[14] D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester and A. Dimou, Declarative RDF Graph Generation from Heterogeneous (Semi-)Structured Data: A Systematic Literature Review, *Journal of Web Semantics* **75** (2023), 100753. doi:10.1016/j.websem.2022.100753.

[15] Knowledge Graph Construction Community Group, 2021.

[16] A. Iglesias-Molina, D. Van Assche, J. Arenas-Guerrero, B. De Meester, C. Debruyne, S. Jozashoori, P. Maria, F. Michel, D. Chaves-Fraga and A. Dimou, The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF, in: *The Semantic Web – ISWC 2023*, Vol. 14266, T.R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng and J. Li, eds, Springer Nature Switzerland, Cham, 2023, pp. 152–175. ISBN 978-3-031-47242-8 978-3-031-47243-5. doi:10.1007/978-3-031-47243-5_9.

[17] C. Bizer and A. Seaborne, D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs, in: *ISWC2004 (Posters)*, 2004.

[18] A.G. Beltran, R. Albertoni, D. Browning, S. Cox, A. Perego and P. Winstanley, Data Catalog Vocabulary (DCAT) - Version 3, W3C Proposed Reccommendation, W3C, 2024.

[19] E. Korkan, S. Käbisch and M. McCool, Web of Things (WoT) Thing Description 1.1, W3C Recommendation, W3C, 2023.

[20] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López and R. García-Castro, LOT: An Industrial Oriented Ontology Engineering Framework, *Engineering Applications of Artificial Intelligence* **111** (2022), 104755. doi:10.1016/j.engappai.2022.104755.

[21] M. Björklund, J. Schönwälder, P.A. Shafer, K. Watsen and R. Wilton, Network Management Datastore Architecture (NMDA), *Request for Comments*, RFC Editor, 2018. doi:10.17487/RFC8342.

[22] M. Björklund, J. Schönwälder, P.A. Shafer, K. Watsen and R. Wilton, NETCONF Extensions to Support the Network Management Datastore Architecture, *Request for Comments*, RFC Editor, 2019. doi:10.17487/RFC8526.

[23] S. Chávez-Feria, R. García-Castro and M. Poveda-Villalón, Chowlk: From UML-based Ontology Conceptualizations to OWL, in: *The Semantic Web*, P. Groth, M.-E. Vidal, F. Suchanek, P. Szekley, P. Kapanipathi, C. Pesquita, H. Skaf-Molli and M. Tamper, eds, Springer International Publishing, Cham, 2022, pp. 338–352. ISBN 978-3-031-06981-9.

[24] M. Poveda-Villalón, A. Gómez-Pérez and M.C. Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!): An on-Line Tool for Ontology Evaluation, *International Journal on Semantic Web and Information Systems (IJSWIS)* **10**(2) (2014), 7–34.

[25] D. Garijo, O. Corcho and M. Poveda-Villalón, FOOPS!: An Ontology Pitfall Scanner for the FAIR Principles, *International semantic web conference (ISWC) 2021: Posters, demos, and industry tracks* **2980** (2021).

[26] Z. Syed, A. Padia, T.W. Finin, M.L. Mathews and A. Joshi, UCO: A Unified Cybersecurity Ontology, in: *AAAI Workshop: Artificial Intelligence for Cyber Security*, 2016.

[27] D. Brickley and L. Miller, Friend of a Friend (FOAF) Vocabulary Specification, 2014.

[28] FAIRsharing Team, FAIRsharing Record for: Quantities, Units, Dimensions and Types, FAIRsharing, 2015. doi:10.25504/FAIRSHARING.D3PQW7.

[29] D. Garijo, WIDOCO: A Wizard for Documenting Ontologies, in: *The Semantic Web – ISWC 2017*, Vol. 10588, C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange and J. Heflin, eds, Springer International Publishing, Cham, 2017, pp. 94–102. ISBN 978-3-319-68203-7 978-3-319-68204-4. doi:10.1007/978-3-319-68204-4_9.

[30] P.-Y. Vandenbussche, G.A. Atemezing, M. Poveda-Villalón and B. Vatant, Linked Open Vocabularies (LOV): A Gateway to Reusable Semantic Vocabularies on the Web, *Semantic Web* **8**(3) (2016), 437–452. doi:10.3233/SW-160213.

[31] Srl-Labs/Containerlab, 2025.

[32] Openconfig/Kne, 2025.

[33] A. Bierman, M. Björklund, J. Schönwälder, K. Watsen and R. Wilton, YANG Library, *Request for Comments*, RFC Editor, 2019. doi:10.17487/RFC8525.

[34] D. Van Assche and C. Debruyne, BURPing through RML Test Cases, in: *Proceedings of the 5th International Workshop on Knowledge Graph Construction Co-Located with 21th Extended Semantic Web Conference (ESWC 2024), Hersonissos, Greece, May 27, 2024*, D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D. Van Assche, eds, CEUR Workshop Proceedings, Vol. 3718, CEUR-WS.org, 2024.

[35] A. Clemm and E. Voit, Subscription to YANG Notifications for Datastore Updates, Request for Comments, RFC 8641, Internet Engineering Task Force, 2019. doi:10.17487/RFC8641.

[36] P. Bonte, R. Tommasini, E. Della Valle, F. De Turck and F. Ongenae, Streaming MASSIF: Cascading Reasoning for Efficient Processing of IoT Data Streams, *Sensors* **18**(11) (2018), 3832. doi:10.3390/s18113832.

[37] G. Haesendonck, Sitt Min Oo, G. De Mulder, M. Derveeuw, P. Heyvaert, W. Maroy, V. Emonet, Kmhaeren, B. De Meester, D. Van Assche, Thomas and Ajuvercr, RMLio/RMLStreamer: V2.5.0, 2023. doi:10.5281/ZENODO.3887065.

[38] M. Grassi, M. Scrocca, A. Carenini, M. Comerio and I. Celino, Composable Semantic Data Transformation Pipelines with Chimera (2023). doi:10.5281/ZENODO.8020088.

[39] B. Claise, J. Quilbeuf, D. Lopez, D. Voyer and T. Arumugam, Service Assurance for Intent-Based Networking Architecture, *Request for Comments*, RFC Editor, 2023. doi:10.17487/RFC9417.

[40] A. Clemm, J. Medved, R. Varga, N. Bahadur, H. Ananthakrishnan and X. Liu, A YANG Data Model for Network Topologies, *Request for Comments*, RFC Editor, 2018. doi:10.17487/RFC8345.

[41] O. Corcho, D. Chaves-Fraga, J. Toledo, J. Arenas-Guerrero, C. Badenes-Olmedo, M. Wang, H. Peng, N. Burrett, J. Mora and P. Zhang, A High-Level Ontology Network for ICT Infrastructures, in: *The Semantic Web – ISWC 2021*, Vol. 12922, A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P. Barnaghi, A. Haller, M. Dragoni and H. Alani, eds, Springer International Publishing, Cham, 2021, pp. 446–462. ISBN 978-3-030-88360-7 978-3-030-88361-4. doi:10.1007/978-3-030-88361-4_26.