# 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 for constructing knowledge graphs. A prominent mapping language is the Resource Mapping Language (RML), which enables the integration of heterogenous 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 like 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 R2RML or the Resource Mapping Language (RML) is gaining an increasing traction [1]. Thus far, these languages have focused on general purpose data sources like 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 grounds on the YANG data modeling language [2]. To access the data available in YANG-capable network elements, there is a variety of network management protocols that can be used like 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 need to implement these protocols and cope with the intricacies of YANG [6].

Since the advent of the YANG language, the network industry has experienced a rapid evolution, with network vendors adding support for the YANG language and network management protocols in their devices and controllers. This evolution has resulted in the creation of a plethora of YANG data models, contributed by vendors, standards developing organizations, and open-source communities. The collection and integration of these YANG data in a knowledge graph would support new use cases such as network service assurance [7] or network digital twins [8].

But, the flexibility of the YANG language makes the construction of a knowledge graph a challenge. YANG data can be encoded in different formats depending on the network management protocol used to interact with the

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

device. For instance, the NETCONF protocol encodes YANG data in XML format and relies on SSH as the transport protocol, whereas the RESTCONF protocol encodes the data in JSON format and builds on HTTP. In this regard, the integration of YANG data into knowledge graphs is still at an early stage with recent proposals like [9], albeit it assumes that YANG data are ingested through message brokers instead of retrieving the data directly from the devices using the corresponding network protocol.

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 for retrieving 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 for declaring 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 lines of work.

## 2. Related work

The evolution of knowledge graphs and the growing interest in them has led to the development of tools and languages that can facilitate their construction. In this regard, declarative mapping languages provide a scalable and flexible approach for transforming structured and semi-structured data into RDF according to a target ontology. The R2RML language [10] was one of the first initiatives within the W3C, which focused on mapping data from relational databases into RDF. To overcome the limitations of this language and provide support for heterogeneous data sources like JSON, XML, or CSV, the RML language [11] was created as an extension of R2RML.

Driven by the Knowledge Graph Construction (KGC) W3C Community Group[1], the RML language is currently under development arranged into multiple modules. Among them, the RML-IO module[2] focuses on formal representations for describing access to data sources and targets in the creation of knowledge graphs. RML-IO advocates for reusing existing ontologies that describe access to data sources and targets, thus, RML-IO does not limit to specific data sources or targets. In this sense, the community has already pinpointed several relevant ontologies that can be leveraged like D2RQ [12] for accessing relational databases, DCAT [13] for remote files, and the Web of Things [14] for web APIs and data streams like MQTT or Apache Kafka. These are some examples of ontologies that can be used to describe general-purpose data sources in RML mappings, however, when it comes to the scope of network management, there are no ontologies for describing access to YANG data sources that could be reused in RML.

## 3. YANG Server Ontology

The YANG Server Ontology has been developed by following the guidelines defined in the Linked Open Terms (LOT) [15] methodology. LOT is a mature and lightweight methodology for the development of ontologies that embraces the best practices from 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.

[1]https://www.w3.org/community/kg-construct/
[2]https://w3id.org/rml/io/spec

### 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, like NETCONF or RESTCONF, while allowing for 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 specification of ontological requirements has been conducted over several sprints, each one tackling a particular subdomain of knowledge in the ontology. For instance, one sprint was dedicated to the analysis of the concept of YANG server and YANG datastores, other sprints addressed the query and subscription operations, while another sprint focused on collecting requirements specific to the NETCONF protocol. For each sprint, the requirements were captured in the form of natural language statements and stored in a CSV file to ease their processing. The requirements were extracted from interviews with experts from the network industry as well as derived from the analysis of standard specifications [3, 6, 16, 17].

### 3.2. Implementation

Based on the ontological requirements specified in the previous step, this activity has implemented the ontology using a formal language. For the conceptualization, the ontology has been represented in a diagram following the Chowlk notation [18], as depicted 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 improved by using the OOPS! tool [19] to identify common pitfalls and the FOOPS! tool [20] to align with the FAIR principles. In the following subsections, the YANG Server Ontology is explained in detail along with a series of RDF examples.

#### 3.2.1. Server core concepts

The core of the ontology revolves around the concept of a YANG server (`ys:YangServer`), which represents YANG-capable network elements such as devices or controllers. This class includes details for connecting to the YANG server through an endpoint (`ys:endpoint`) expressed in the form *<host>:<port>* along with basic authentication credentials based on username (`ys:username`) and password (`ys:password`). The ontology captures the YANG server concept as a generic class so that future modules can extend the ontology with support for network management protocols such as NETCONF or RESTCONF through subclasses derived from `ys:YangServer`.

On the other hand, the core concept of 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) [16], 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 datastores 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.

#### 3.2.2. NETCONF protocol

Extending from the server core concepts, the ontology 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 for describing 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`). An example describing a NETCONF server is depicted in Listing 1.

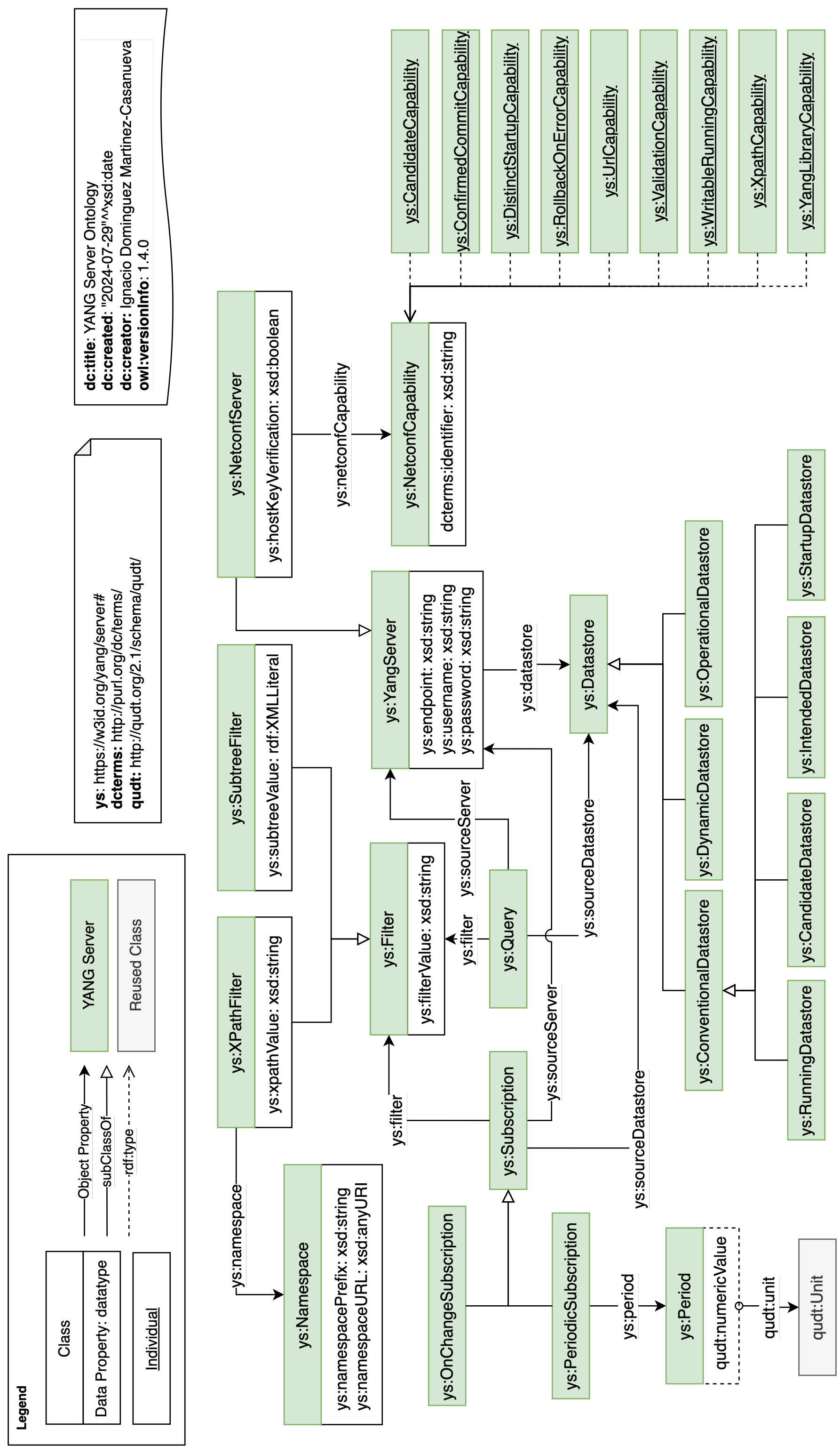

Fig. 1. Conceptual representation of YANG Server Ontology (Represented using the Chowlk visual notation [18]).

Listing 1: NETCONF server represented in Turtle format.

```
@prefix ys: <https://w3id.org/yang/server#> .
@base <https://example.org/> .

<netconf-server-1> a ys:NetconfServer ;
    ys:endpoint "192.168.1.10:830" ;
    ys:username "admin" ;
    ys:password "admin" ;
    ys:hostKeyVerification "false" ;
    ys:datastore <datastores/netconf-server-1/operational> ;
    ys:datastore <datastores/netconf-server-1/running> ;
    ys:capability ys:XpathCapability .

# In this example the NETCONF server is non-NMDA compatible.
# Thus, only operational and running datastores are available.
<datastores/netconf-server-1/operational> a ys:OperationalDatastore .
<datastores/netconf-server-1/running> a ys:RunningDatastore .
```

### *3.2.3. YANG operations*

Network management operations for retrieving data from a YANG server are covered by the ontology. In particular, queries (`ys:Query`) and subscriptions to notifications (`ys:Subscription`) have been defined. In a query, a single request is sent to the server to fetch 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.

The creation of a query requires indicating the server and 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 (see Listing 2). The subtree filter offers a fine-grained mechanism to select the data to be filtered and returned (see Listing 3). 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).

Listing 2: Query with XPath filter to the operational datastore of a NETCONF server.

```
<query-xpath> a ys:Query ;
    ys:sourceServer <netconf-server-1> ;
    ys:sourceDatastore <datastores/netconf-server-1/operational> ;
    ys:filter [ a ys:XPathFilter ;
        ys:xpathValue "/yanglib:modules-state/yanglib:module" ;
        ys:namespace [ a ys:Namespace ;
            ys:namespacePrefix "yanglib" ;
            ys:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-yang-library" ;
        ];
    ];
.
```

Listing 3: Query with Subtree filter to the running datastore of a NETCONF server.

```
<query-subtree> a ys:Query ;
    ys:sourceServer <netconf-server-1> ;
    ys:sourceDatastore <datastores/netconf-server-1/running> ;
    ys:filter [ a ys:SubtreeFilter ;
        ys:subtreeValue '''
        <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
          <interface>
          </interface>
        </interfaces>
        ''';
    ];
.
```

Regarding YANG subscriptions, a server, datastore, and filter must be specified, but also additional aspects that will depend on the type of subscription. A subscription may trigger notifications upon updates on the data selected by the filter (`ys:OnChangeSubscription`). Listing 4 shows an example of a subscription that triggers notifications when the "enabled" status of a network interface changes. Alternatively, a subscription may trigger notifications periodically based on the specified time interval (`ys:PeriodicSubscription`). In this case, the QUDT Ontology [21] has been leveraged to represent the interval used by the subscription. As an example, Listing 5 represents a periodic subscription that pushes the value of the output octets counter of the network interfaces every 10 seconds.

Listing 4: On-change subscription with XPath filter to the running datastore of a NETCONF server.

```
<sub-onchange> a ys:OnChangeSubscription ;
    ys:sourceServer <netconf-server-1> ;
    ys:sourceDatastore <datastores/netconf-server-1/running> ;
    ys:filter [ a ys:XPathFilter
        ys:xpathValue "/if:interfaces/if:interface/if:enabled" ;
        ys:namespace [ a ys:Namespace ;
            ys:namespacePrefix "if" ;
            ys:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-interfaces" ;
        ];
        ys:namespace [ a ys:Namespace ;
            ys:namespacePrefix "ip" ;
            ys:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-ip" ;
        ];
    ];
.
```

Listing 5: Periodic subscription with XPath filter to the operational datastore of a NETCONF server.

```
<sub-periodic> a ys:PeriodicSubscription ;
    ys:sourceServer <netconf-server-1> ;
    ys:sourceDatastore <datastores/netconf-server-1/operational> ;
    ys:filter [ a ys:XPathFilter ;
        ys:xpathValue "/if:interfaces-state/if:interface/if:statistics/if:out-octets" ;
        ys:namespace [ a ys:Namespace ;
            ys:namespacePrefix "if" ;
            ys:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-interfaces" ;
        ];
        ys:namespace [ a ys:Namespace ;
            ys:namespacePrefix "ip" ;
            ys:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-ip" ;
        ];
    ];
    ys:period [ a ys:Period ;
        qudt:numericValue "10" ;
        qudt:unit <https://qudt.org/2.1/vocab/unit/SEC>
    ];
.
```

### *3.3. Publication*

This activity focuses on documenting and making the ontology publicly available. The documentation has been generated using the WIDOCO tool [22] and the URL of the ontology has been registered under the W3ID domain[3]. The ontology artifacts, such as requirements specification, diagram, code, or documentation, are available in a GitHub repository[4]. Additionally, the ontology has been registered in the Linked Open Vocabulary (LOV) service [23] to improve its discoverability.

[3]http://w3id.org/yang/server/
[4]https://github.com/candil-data-fabric/yang-server-ontology

### *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 the YANG Server Ontology

When aligning the YANG Server Ontology with RML to represent YANG servers as logical sources, 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 under the same server, therefore, the retrieval of data happens on a particular YANG datastore. But, in addition to indicating the source datastore, it should be possible to include a filter in the form of an XPath expression or XML subtree to filter out data at the server. From the perspective of the RML mappings, filtering out data at the server 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. For these reasons, YANG operations (i.e., `ys:Query` and `ys:Subscription`), which indicate the source server (`ys:YangServer`), the source datastore (`ys:Datastore`), and the filter (`ys:Filter`), are identified as the sources (`rml:Source`) in the RML mappings.

However, the type of filter used in the operation as well as the protocol implemented by the server will determine the final representation of the logical source in the RML mappings. For instance, the NETCONF protocol encodes YANG data in XML format, thus, the RML mappings will need to be defined to iterate over XML data.

In this sense, Listing 6 depicts an example of a NETCONF server accessed with a query containing an XPath filter. In this case a filter (`ys:Filter`) is not needed in the RML mapping since the XPath expression (`rml:iterator`) and the definition of XML namespaces (`rml:Namespace`) are used for iterating over the data. Note that the use of `rml:XPathReferenceFormulation` is mandatory because XPath expressions in NETCONF must always be prefix-aware.

Listing 6: RML logical source represented by a YANG Query with XPath filter to a NETCONF server.

```
rml:logicalSource [ a rml:LogicalSource;
  rml:source [ a ys:Query, rml:Source ;
    ys:sourceServer <netconf-server-1> ;
    ys:sourceDatastore <datastores/netconf-server-1/running> ;
  ];
  rml:referenceFormulation [ a rml:XPathReferenceFormulation;
    rml:namespace [ a rml:Namespace ;
      rml:namespacePrefix "if" ;
      rml:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-interfaces" ;
    ];
    rml:namespace [ a rml:Namespace ;
        rml:namespacePrefix "ip" ;
        rml:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-ip" ;
    ];
  ];
  rml:iterator "/if:interfaces/if:interface";
];
```

On the other hand, when the query to a NETCONF server includes an XML subtree filter, the representation of the logical source slightly differs, as shown in Listing 7. In this case, the `ys:SubtreeFilter` is included to filter the data in the server, and later, `rml:XPathReferenceFormulation` and the XPath expression (`rml:iterator`) are specified to indicate how the RML engine must iterate over the XML-encoded data.

Listing 7: RML logical source represented by a YANG Query with XML subtree filter to a NETCONF server.

```
rml:logicalSource [ a rml:LogicalSource;
  rml:source [ a ys:Query, rml:Source ;
    ys:sourceServer <netconf-server-1> ;
    ys:sourceDatastore <datastores/netconf-server-1/operational> ;
    ys:filter [ a ys:SubtreeFilter ;
      ys:subtreeValue '''
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
        </interface>
      </interfaces>
      ''';
    ];
  ];
  rml:referenceFormulation [ a rml:XPathReferenceFormulation;
    rml:namespace [ a rml:Namespace ;
      rml:namespacePrefix "if" ;
      rml:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-interfaces" ;
    ];
    rml:namespace [ a rml:Namespace ;
        rml:namespacePrefix "ip" ;
        rml:namespaceURL "urn:ietf:params:xml:ns:yang:ietf-ip" ;
    ];
  ];
  rml:iterator "/if:interfaces/if:interface";
];
```

Additionally, note that these RML mapping examples have been illustrated by queries, but an equivalent approach would be followed for subscriptions.

## 5. Use case: evolution of the YANG Catalog

YANG Catalog[5] is a web service that provides a search tool for finding existing YANG modules. The YANG Catalog service provides additional metadata like 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 module metadata in a graph structure, the traversal of the different types of dependencies among modules is optimized; and ii) the semantic layer built with the knowledge graph enables integrating the module metadata with other silos of data. The second aspect in particular proves the benefits of knowledge graphs as technology for breaking silos of data. For example, the metadata can be integrated with data related to the network topology [24], providing network operators with insights about which modules are implemented by the devices in the network. Similarly, module metadata could also be linked in the knowledge graph with network concepts captured in formal vocabularies, enabling searches such as: *Which YANG modules refer to the "Interface" or "IPv6" concepts?*.

To create a knowledge graph that supports the YANG Catalog, the proposed approach builds upon collecting the YANG Library data from devices running in the network. YANG Library [25] is a YANG module that provides information about the YANG modules and datastores used by a YANG server. In this regard, the RML engine responsible for creating the knowledge graph interacts with the network device via a protocol like NETCONF to ingest the YANG Library data and, based on the declared mappings to the target ontology, integrates the data in the knowledge graph. To validate this solution for the use case, the prototype depicted in Fig. 2 has been developed.

The netopeer2[6] tool has been used to emulate a NETCONF server, which has been configured to expose the YANG Library[7] data. The selected RML engine has been BURP [26], which is being developed under the um-

[5]https://www.yangcatalog.org
[6]https://github.com/CESNET/netopeer2
[7]https://github.com/network-analytics/libyangpush

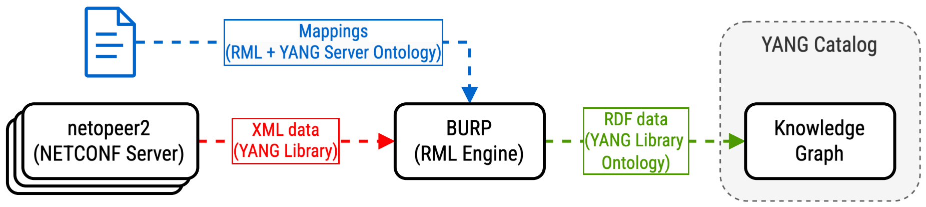

Fig. 2. Prototype overview. Based on the mappings, BURP obtains the YANG Library data from the NETCONF servers using the NETCONF protocol. The RDF data generated by BURP are stored in a knowledge graph, which acts as the database of the YANG Catalog service.

brella of the KGC Community Group as the reference implementation compliant with the latest RML specification. For this prototype, BURP has been extended[8] with two major features: i) capability to process the `rml:XPathReferenceFormulation` to iterate over XML data using an XPath expression along with a map of XML namespaces and prefixes; and ii) support NETCONF sources as logical sources in RML based on the YANG Server Ontology. Finally, example mappings for integrating the YANG Library data into the knowledge graph have been developed, referencing the YANG Library Ontology[9]. Examples of the input YANG Library data, the mappings, and the generated RDF data have also been uploaded to the BURP repository.

## 6. Conclusions and future work

This paper has introduced the YANG Server Ontology, which allows for representing YANG servers and network management operations with YANG servers. The ontology has been developed following a well-known, mature 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 referenced by the RML language for declaring YANG servers 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.

A possible future contribution could focus on validating YANG subscriptions. Servers implementing this feature can improve the construction of the knowledge graph on a streaming basis, thus, unlocking real-time use cases. The proposed YANG Server Ontology has addressed the concept of YANG subscriptions but its combination with RML has not been implemented yet. In comparison to the common batch approach followed by the existing RML engines whereby data is pulled from the data source, this feature would need to introduce a new paradigm in RML engines in order to create a session with the data source (e.g., NETCONF server) through which the notifications are received.

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 upon the gRPC protocol[10] and the Protocol Buffers[11] 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 this 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).

[8]https://github.com/kg-construct/BURP/pull/5
[9]http://w3id.org/yang/library/
[10]https://grpc.io
[11]https://protobuf.dev

## 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.

[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] 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.

[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] 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.

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

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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).

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.