μRDF Store: Towards Extending the Semantic Web to Embedded Devices

Victor Charpenay[1,2](✉), Sebastian Käbisch[1], and Harald Kosch[2]

[1] Siemens AG — Corporate Technology, Munich, Germany
{victor.charpenay,sebastian.kaebisch}@siemens.com
[2] Fakultät für Informatik und Mathematik, Universität Passau, Passau, Germany
harald.kosch@uni-passau.de

Abstract. This paper presents the μRDF store, a triple store designed for micro-controllers with limited memory, typically 8 to 64 kB. The μRDF store exposes a query interface inspired by SPARQL that supports basic graph pattern queries. Data is sent over CoAP and serialized in EXI, a binary format for XML.

The performances of its processing engine are demonstrated in a Web chat application where the μRDF store can be submitted queries. The application is available at: https://vcharpenay.github.io/urdf-amaa/.

Keywords: Web of things · Internet of things · SPARQL · RDF · EXI · CoAP

1 Introduction

In the past few years, Semantic Web technologies have been successfuly applied to the domain of the Internet of Things (IoT). Various systems such as SPITFIRE [6], mixing RDF and machine-generated data, were developed with promising results.

In this paper, we explore the possibility of exchanging RDF data in constrained environments, in order to extend the scope of the Semantic Web for the IoT. In particular, we are interested in the problem of storing and querying RDF data on micro-controllers with IP connectivity (8 to 64 kB RAM).

2 Related Work

Until recently, no realistic use case could be found where computational devices had limited resources but still IP connectivity. As a consequence, the problem of storing RDF in constraind environments, as opposed to storing billions of statements in very large databases, has remained mostly unexplored. The situation has changed with the coming of the IoT where RDF has found new usages.

The first work that addressed constrained devices —and to the best of our knowledge, the only one— is the Wiselib TupleStore (as part of SPITFIRE) [4].

It compares the performance of various C++ data structures to compress URIs, inherent to any RDF document.

The Wiselib TupleStore features insertion and removal operations. There exists other proposals to compress RDF data but none of them have these characteristics. The most notable ones are the Header Dictionary Triples (HDT) binary representation for RDF [2] and k^2-triples, a variant of HDT [1]. The main objective of HDT was to compress large RDF datasets. e.g. to fit in the main memory of a standard PC. But its compression scheme could reasonably be used on small datasets as well.

In the original proposal for HDT, triples are stored using bitmaps on which compression is applied. k^2-triples indexes triples in subject-object indexes (vertical partitioning) and applies k^2-tree compression on the two-dimensional arrays.

HDT and k^2-triples show high compression ratios. However, in a typical IoT configuration, RDF data might be dynamically updated, which requires insertion and removal operations. Moreover, neither the Wiselib Tuplestore nor HDT/k^2-triples go beyond triple pattern matching to query RDF. Our proposal, the μRDF store, combines both a small memory footprint with insertion/removal operations and Basic Graph Pattern (BGP) processing. It is presented next.

3 Overview of the μRDF Store

A few assumptions were made in the design of the μRDF store. First, it is expected that datasets contain a limited number of triples, to fit in the RAM of a micro-controller. Second, we expect datasets to contain mostly assertional data, as opposed to terminological data.

3.1 Data Structure

The data structure underlying the μRDF store is optimized for navigational queries (i.e. with bound predicates) against datasets having few distinct properties (max. n properties). Each resource is given an index number where the first n indices are reserved for predicates (in practice, $n = 32$ or 64, i.e. a multiple of the size of a machine word). Triples are stored with variable-length byte (vbyte) encoding in a way that navigation from every resource to its neighbors in the RDF graph is possible both forward and backward. Triples with literals are only stored once.

Formally, for I, B and L mutually disjoint sets of IRIs, blank nodes and literals respectively, a μRDF resource $\rho \in P$ is the pair (R, L) where $R \subset I \times (I \cup B) \times \{0, 1\}$ is the set of direct and reverse relations to other resources and $L \subset I \times L$ is the set of literal relations associated to ρ. An instance of the μRDF store is a map $\mu : (I \cup B) \rightarrow P$.

3.2 Query Interface

The μRDF store supports BGP queries (given that at least one subject or object is bound). Queries are processed in a greedy fashion: as soon as a binding is found for a triple pattern, the next triple pattern is processed. This guarantees an upper bound of $O(|V|)$ for intermediate storage, where V is the set of variables in the query. This process, that might lead to duplicated calls, avoids managing arbitrarily large intermediary results (e.g. for patterns with the `rdf:type` predicate), which is critical in constrained environments.

To exchange data over the network, data is serialized in the Efficient XML Interchange (EXI) format. Our original proposal for the μRDF store suggests that EXI, that can take advantage of schema information from RDF/XML, offers satisfactory compression ratios [5].

Moreover, a straightforward mapping of the HTTP SPARQL 1.1 protocol can be defined for the Constrained Application Protocol (CoAP), where both query and update are supported. The CoAP specification limits the payload size to 1024 bytes. If the serialized result set is above this value, the algorithm described above allows data to be sent "block-wise": each full mapping found by the algorithm is sent in a separate block[1].

4 Evaluation

We implemented the μRDF store for the ESP8266, a microcontroller with an integrated Wi-Fi chip (64 kB RAM, 80 MHz). To test its performances, we used the dataset generator and the query mix provided by the Lehigh University Benchmark (LUBM) [3]. The generator had to be adapted to small datasets[2]. All fourteen queries of the benchmark can be processed by our implementation.

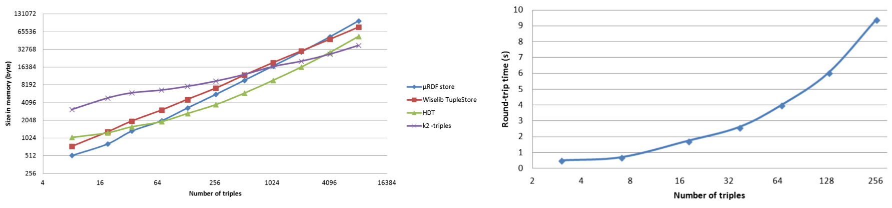
We compared the memory footprint of our implementation to the Wiselib tuplestore, HDT and k^2-triples (Fig. 1a). The comparison suggests that, overall, HDT performs best, followed by the μRDF store, in the range we consider of interest (up to 10,000 triples). It is reported in the literature that k^2-triples performs better but the graphic shows that this does not apply to small RDF datasets. It also shows that URI compression as performed by the Wiselib tuplestore performs poorly for datasets with less than 1,000 triples.

The original idea behind the μRDF store is to explore SPARQL as a decentralized discovery mechanism among IoT devices. This requires at least BGPs to be supported by individual nodes. As mentioned before, no work in the state-of-the-art has proposed an implementation of BGP processing at the scale of a micro-controller.

On the MSP8266, all queries from the LUBM benchmark —especially Q9 that requires many intermadiary joins—are processed in less than 20 ms, including EXI coding. In comparison, sending static RDF/EXI data over CoAP takes at least 500 ms and this number grows with the number of triples being sent (Fig. 1b), which means that query processing has no significant impact on the overall exchange between two devices. On the contrary, local BGP processing,

[1] The demonstration presented in this paper also exploits this partitioning of the result set with MQTT, another IoT protocol: each mapping is published in a separate MQTT message.

[2] See https://github.com/vcharpenay/urdf-store-exp for more details.



(a) Memory footprint

(b) CoAP round-trip time (RTT)

Fig. 1. Evaluation of the μRDF store with LUBM synthetic data

in constrat to other RDF query mechanisms such as Triple Pattern Fragments (TPFs) [7], can significantly improve the rapidity of an exchange by reducing the total amount of exchanged data.

Our work on the μRDF store has led us to the following conclusion: with today's hardware, an "Embedded Semantic Web" is possible. As opposed to HDT, the μRDF store supports updates and BGP processing; without degrading data transmission; with a limited overhead in memory.

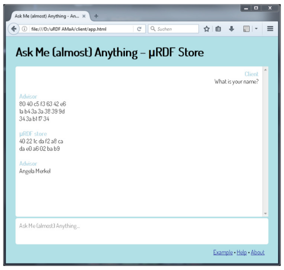
5 Demonstration: Ask Me (almost) Anything

The purpose of this demonstration is to illustrate the conclusions of our evaluation in an interactive fashion. We developed a simple chat application that allows one to submit queries in a simplified manner to an instance of the μRDF store from their Web browser. Its design is inspired by a concept popularized by Reddit called Ask Me Anything (AMA). Here, a microcontroller invites you to an AMA session to let you discover its name, its capabilites and the "Things" it is semantically connected to.

Figure 2a shows the Web interface on which queries can be formulated. The technical details of the demonstration are given in Fig. 2b. A client query is first pre-processed by the so-called "advisor", a non-constrained machine that first translates the query into SPARQL, then serializes it in EXI; the EXI is processed by the micro-controller which responds with another EXI message; the server response is post-processed by the advisor to print it in a human-readable form. All the messages are sent via the Message Queue Telemetry Transport (MQTT) protocol following a simple publish-subscribe interaction pattern. The delay between requests and responses does not exceed a few seconds.

6 Conclusion

IoT devices that embed a full IP-based communication stack usually come with unused computational power. As current applications mostly concentrate on sending sensor data to the Cloud as fast as possible, the capacities of edge devices remain untapped. With the μRDF store, we aim at relocating intelligence to the



Ask Me (almost) Anything - An...

file:///D:/uRDF AMaA/client/app.html

Ask Me (almost) Anything – µRDF Store

Client
What is your name?

Advisor
80 40 c5 f3 63 42 e6
1a b4 3a 3a 38 39 9d
34 3a b1 17 34

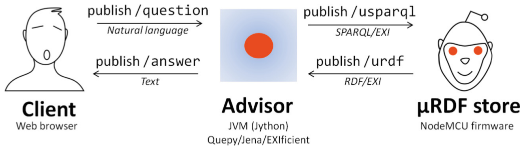
µRDF store
40 22 1c da f2 a8 ca
da e0 a6 02 ba b9

Advisor
Angela Merkel

Ask Me (almost) Anything...

Example • Help • About

(a) Web client screenshot



(b) MQTT message exchange

Fig. 2. Overview of the ask me (almost) anything (AMaA) application

edge, starting with semantic processing. Our demonstration underlines the relevance of an Embedded Semantic Web for the IoT.

Another aspect that this demonstration tends to show is that human–to–machine communication is tedious. IoT systems might be much more performant if machines themselves formulated the queries. This is our next line of research. Support for the `OPTIONAL` operator is also being considered in our implementation.

References

1. Álvarez-García, S., Brisaboa, N.R., Fernández, J.D., Martínez-Prieto, M.A.: Compressed k2-triples for full-in-memory RDF engines. CoRR, abs/1105.4004 (2011)
2. Fernández, J.D., Martínez-Prieto, M.A., Arias, M., Gutierrez, C., Álvarez-García, S., Brisaboa, N.R.: Lightweighting the web of data through compact RDF/HDT. In: Lozano, J.A., Gómez, J.A., Moreno, J.A. (eds.) Advances in Artificial Intelligence. volume 7023, pp. 483–493. Springer, Berlin (2011)
3. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. Web Semant. Sci. Serv. Agents World Wide Web **3**(23), 158–182 (2005)
4. Hasemann, H., Kröller, A., Pagel, M.: The wiselib tuplestore: a modular RDF database for the internet of things. CoRR, abs/1402.7228 (2014)
5. Käbisch, S., Peintner, D., Anicic, D.: Standardized and efficient RDF encoding for constrained embedded networks. In: Gandon, F., Sabou, M., Sack, H., dAmato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 437–452. Springer, Cham (2015). doi:10.1007/978-3-319-18818-8_27
6. Pfisterer, D., Romer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., Hasemann, H., Kröller, A., Pagel, M., Hauswirth, M., Karnstedt, M., Leggieri, M., Passant, A., Richardson, R.: SPITFIRE: toward a semantic web of things. 49(11), pp. 40–48 (2011)
7. Verborgh, R., Hartig, O., Meester, B., Haesendonck, G., Vocht, L., Vander Sande, M., Cyganiak, R., Colpaert, P., Mannens, E., Walle, R.: Querying datasets on the web with high availability. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 180–196. Springer, Cham (2014). doi:10.1007/978-3-319-11964-9_12