A Preliminary Open Data Publishing Strategy for Live Data in Flanders

Julián Andrés Rojas Meléndez
Ghent University – imec – IDLab
julianandres.rojasmelendez@ugent.be

Brecht Van de Vyvere
Ghent University – imec – IDLab
brecht.vandevyvere@ugent.be

Arne Gevaert
Ghent University – imec – IDLab
arne.gevaert@ugent.be

Ruben Taelman
Ghent University – imec – IDLab
ruben.taelman@ugent.be

Pieter Colpaert
Ghent University – imec – IDLab
pieter.colpaert@ugent.be

Ruben Verborgh
Ghent University – imec – IDLab
ruben.verborgh@ugent.be

ABSTRACT

For smart decision making, user agents need live and historic access to open data from sensors installed in the public domain. In contrast to a closed environment, for Open Data and federated query processing algorithms, the data publisher cannot anticipate in advance on specific questions, nor can it deal with a bad cost-efficiency of the server interface when data consumers increase. When publishing observations from sensors, different fragmentation strategies can be thought of depending on how the historic data needs to be queried. Furthermore, both publish/subscribe and polling strategies exist to publish live updates. Each of these strategies come with their own trade-offs regarding cost-efficiency of the server-interface, user-perceived performance and cpu use. A polling strategy where multiple observations are published in a paged collection was tested in a proof of concept for parking spaces availability. In order to understand the different resource trade-offs presented by publish/subscribe and polling publication strategies, we devised an experiment on two machines, for a scalability test. The preliminary results were inconclusive and suggest more large scale tests are needed in order to see a trend. While the large-scale tests will be performed in future work, the proof of concept helped to identify the technical Open Data principles for the 13 biggest cities in Flanders.

KEYWORDS

Open Data, Live data, Linked Data, REST

ACM Reference Format:
Julián Andrés Rojas Meléndez, Brecht Van de Vyvere, Arne Gevaert, Ruben Taelman, Pieter Colpaert, and Ruben Verborgh. 2018. A Preliminary Open Data Publishing Strategy for Live Data in Flanders. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 7 pages. *https://doi.org/10.1145/3184558.3191650*

1 INTRODUCTION

Live open datasets, in this paper, are defined as datasets that can be modeled using a ledger-like data model, as illustrated in fig. 1. Examples would be sensor observations in the public domain about e.g., air quality, noise level, street occupancy, vacant parking spaces, temperature, river water level, wind speed or state of public lighting systems and traffic lights. Furthermore, also slower changing data can be modeled as live data, for example, administrative data published by public services, such as the local decisions of a city council [3], wanting to reduce the latency between updated legislation and user advice provided by third party interfaces. Potentially, also predictions of future, or planning data can be published, such as with public transport timetables [6] or occupancy predictions [17].

Figure 1: Live or real-time data are data modeled by "observations" (black ticks). A live update is when new observations come into play. Older data is historic data. Optionally, planned data or predictions can be published.

Publishing data for maximum reuse – the goal behind open data – is particularly different than sharing data between only two parties. First, one cannot anticipate on specific questions: any question can be asked with an *open world assumption*. Second, it becomes too expensive for both consumer to reuse the data, as for the publisher to share data, if there is a manual negotiation process that needs to take place. Therefore, Open Data publishing is an automation challenge on a legal, technical, syntatic, semantic and querying level [5].

Legally, open data licenses are described as part of the Open Definition[1]. Technically, the HTTP protocol can be used to bring datasets physically together. Syntactically, the RDF 1.1 w3c specification standardizes open formats for all data that can be modeled in triples. Semantically, using HTTP URIs avoids conflicting identfiers and makes the meaning of the identifier accessible by doing a simple lookup. Data publishers can reuse URIs from existing domain models, making the data more useful to a broader audience. Using the HTTP protocol as a uniform interface, *content negotiation* can deliver different serializations for the same information resources, making the identifiers technically interoperable over software systems. Furthermore, also profiles [18] can be negotiated, giving a more fine-grained choice of which domain model is preferred by a user agent.

[1] *http://opendefinition.org*

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191650

In order for clients to be able to query over different sources, we propose to look for a minimal contract between the server and the client. The looser this contract, the higher the probability that user agents can consume another dataset without modifying its code; hence, lowering the cost for adoption of heterogeneous data on the Web. In this paper we design such information system by proposing a set of requirements that follow the REST constraints and aim for maximizing reusability of live open datasets and their historic observations. Furthermore, we present a proof of concept of live open data on the Web, portraying public parking availability of several cities in Flanders. Finally, in order to have a clearer picture of the cost-efficiency trade-offs inherent to publishing live open data on the Web, we propose an experiment to study different Web interface setups that follow polling or publish-subscribe strategies.

The contributions of this paper are:

(1) We propose a minimum set of technical requirements to publish live data over HTTP for the 13 biggest cities in Flanders
(2) HTTP polling and publish/subscribe based approaches such as Websockets can be used to publish open data streams on the Web. We designed a preliminary experiment in order to find out under what conditions is more suitable to select one approach or the other for stream open data publishing.

The remainder of the article is structured as follows. We first describe the state of the art, we then list the most important needs in an Open Data context and then we describe the proposed experiment to evaluate the cost-efficiency of different Web interfaces and present a set of preliminary results. Afterwards, we describe a proof of concept implementation for the proposed system in Flanders. Finally we present conclusions and directions for future work.

2 STATE OF THE ART

Representational State Transfer (REST) is a set of constraints, derived while standardizing HTTP/1.1 [9]. The *uniform interface constraint* requires that every individual information resource on the Web is accessed through a single identifier – an HTTP URI – regardless of the specific format chosen to represent it. These identifiers can be used to link information resources together through *hypermedia controls* such as links or web forms. This allows user agents to discover new information resources while following their nose. Also for non-information resources, such as real-world objects, the *Linked Data* principles advocate for using similar HTTP URIs, allowing a uniform strategy to look up the definitions of identifiers, and avoiding conflicting identifiers. Other constraints were formulated such as the caching constraint, client-server constraint or the statelessness constraint. As with any architectural style, developers can choose to follow these constraints, or to ignore them. By following these constraints, REST promises beneficial properties, such as network efficiency, scalability, reliability, a good user-perceived performance and simplicity. The REST architectural constraints do not further specify how a specific API should be implemented. To that extent, the World Wide Web Consortium (w3c) has published a list of best practices for data on the Web[2]. The 35 best practices are generic publishing guidelines for data. They have been extentended in collaboration with the Open Geospatial Consortium (OGC), to also create the "Spatial Data on the Web Best Practices"[3]. Finally, Hydra, a community group at the w3c, is a vocabulary of URIs that allow for describing hypermedia controls in Linked Data documents.

RDF streams [7] are a popular way for representing live Linked Data data on the Web. VoCaLS[4] introduced a vocabulary and ontology that aims to describe RDF streams and access points for publishing and consuming such streams. TripleWave [12] is an approach in which RDF streams can be published by mapping raw data streams to RDF and publishing them based on the Linked Data principles. While TripleWave does allow for mapping streaming data to Linked Data on the fly, it does not yet provide a strategy to store nor access historic data.

This approach in this paper is based on the concepts of *Stream Graphs*(sGraphs) and *Instantaneous Graphs*(iGraphs) [2], which are both *named graphs* [4]. In this approach a RDF stream corresponds to a sGraph, which references iGraphs. Each iGraph is annotated with a timestamp that represents the timestamp of each RDF statement in that graph. Using this model we group observations in Linked Data documents to provide access to historic stream data. This contrasts with the RDF Stream Processing (RSP) community, which mainly focuses on the last updates. This idea is described further in Chapter 4.

RSP engines [1, 11] can be used for making RDF streams queryable using an extended SPARQL [10] syntax. TPF-QS [15] was introduced as an alternative to server-side RSP engines with the goal of making RDF stream server-side publishing possible at a low cost, with a client-side RSP engine. In this approach, several time-annotation techniques were investigated, of which annotation using named graphs caused the least overhead. The results however showed that this approach has scalability issues when querying historic data [15]. In this paper we do not go further into these query engines, as a good data publishing strategy should enable third parties to use the data in their query engines. Therefore, we do not see a query evaluation endpoint as a core task of a data publisher.

Finally, on a Web of Data Streams introduced 7 requirements [8] for data streams in order to make streaming data reusable. These requirements, created independently around the same time, show parallels with our requirements introduced in the next chapter.

3 LIVE DATA ON THE WEB

As a result of one year of technically assisting 13 cities in publishing live parking availabilities, we introduce seven requirements for the automation of adopting live open data in third party interfaces. These requirements show similarities with related work [8, 13], yet differ where cost-efficiency for open datasets published by municipalities is needed.

1. Large heterogeneity of questions When a large time series is published, any kind of analysis must be possible.

2. A highly available interface These sources must be *highly available*, as downtime from the data source means the end-user will experience a direct harm of having to work with out-dated data, or even not being able to answer certain questions. On the one hand high availability may be achieved with failover backup systems, resource sharing, clustering

[2] *https://www.w3.org/TR/dwbp/*

[3] *https://www.w3.org/TR/sdw-bp/*

[4] *https://w3id.org/rsp/vocals#*

techniques, among others. However, these solutions mean increased costs for the data publisher. On the other hand, reducing the server-side processing responsibilities may increase the availability while improving the scalability and the cost-efficiency of the data interface.

3. **Low latency for data responses** For live data, the latency between an update registered on the data publishing server and the use in user agents needs to be minimized. Data publishing servers shall be able to process high-volumes of streaming data with low latency. Therefore, a stream data publishing server must have a highly optimized, minimal-overhead execution to deliver live responses to a high volume of clients [13].
4. **Automatic discovery of legal constraints** The Open Definition allows attribution or share alike constraints. A user agent should be able to automatically discover what attribution or share alike constraints will be applicable on the resulting response.
5. **Discoverable information resources** A user agent should be able to discover all possible referenced sources by following links. Once data sources are discovered, questions should be able to be evaluated. This way, all pages can be discovered, but also server-side features can be turned off and on at will. Also the URLs used will be generated by the server, and will be shared across all possible user agents, delivering a higher possibility to cache resources in between. Leading to a higher cacheability, and thus a higher user-perceived performance as a more cost-efficient interface.
6. **Cross origin resource sharing** As these datasets are going to be published on the Web, also different pages with client-side code need to be able to access resources from different domains. This is by default disabled for security reasons.
7. **Access to historic data** In order to make statistical analysis or create prediction models, access to the history is required.

We propose these 7 requirements to maximize reusability of live open data streams on the Web. As previously mentioned in the state of art chapter, by comparing our proposed requirements with the ones proposed in the work "On a Web of Data Streams" [8] we can identify several similarities especially from a reusability perspective (R5, R6 and R7). However, we particularly differ on the "Keep Data Moving" requirement (R1) where is stated that data streams should use a non-polling approach. We argue that using a polling approach and particularly due to its caching capabilities, it is possible to increase the availability, scalability and thus the cost-efficiency of open data stream interfaces on the Web. However, we do not exclude non-polling publication strategies as we devise a trade-off of using one approach or the other, that should be determined by the data publisher depending on its constraints and the use case. Moreover, as we detail on the next chapter, we provide also access to stream historic data, which is not portrayed as a requirement in [8] and is vital to provide answers to more complex queries.

In the next chapter we propose a way to publish the live and historic data in the same interface, that splits a data dump in fixed fragments. The latter will allow each fragment to have a HTTP URL, caching headers can be installed on the responses, content negotiation can be supported, and other HTTP features can be exploited. Links and hypermedia controls in between these information resources aid the discoverability.

4 A LOOSE CONTRACT FOR OPEN TIME SERIES DATA

Instead of creating a new information resource for each new "version" or observation, we propose to group observations in information resources. Named graphs in the RDF 1.1 specification can be used to extend triples with specific content, such as provenance data, explaining when a certain piece of data was generated. Fragments of older data have the interesting property that they will not change any longer and thus become cacheable for a long time. The only document that might change more often, is the document containing the current time. As HTTP is our uniform interface, we must set the HTTP cache headers according to this specification.

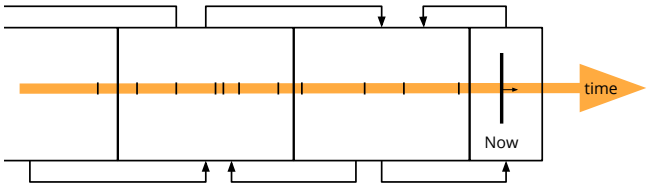

Figure 2: When publishing data in fragments, a generic fragmentation strategy can be thought of for "real-time" open data.

As illustrated in fig. 2, one document will refer to other pages. Following requirement 5, we must provide links between these different documents in order the find the previous or next page. The Hydra vocabulary (*http://hydra-cg.com*) can do this in the body of an HTTP response, and also the Link header can provide this functionality.

We can see this approach as part of the Linked Data Fragments axis [19] where it holds great similarities with a Linked Data Documents based solution. Located towards the far left side of the axis, it is characterized by presenting low server costs by entrusting demanding tasks such as query resolution to the clients, which is essential for open data publishers. In contrast, RSP based approaches could be located towards the far right side of the axis where server interfaces have higher costs, as most of the processing tasks are performed in the server. A solution of this kind for open data on the Web, where the number of users cannot be anticipated, may create scalability issues (requirement 3) in limited infrastructures. Entrusting query resolution tasks to the clients to lower the cost of server interfaces for streaming data was demonstrated in [15].

As user agents need to be able to ask questions over different sources, the terms that are used to describe certain observations must be globally unique and persistent. As the Web is our uniform interface, we choose web addresses or URIs for our identification strategy. This enables user agents to semantically integrate datasets published by different sources. For the domain models, we recommend reusing existing vocabularies. An overview of reusable vocabularies is available at *http://lov.okfn.org* [16].

Web browsers are the ultimate HTTP client. End-user pages contain scripts which may make decisions depending on the end-user's

context such as geo-location. For security reasons however, these scripts cannot access datasets on other domains by default. In order to allow this, the HTTP responses must contain a cross origin resource sharing header.

Summarizing, for adding "real-time" open data to the Web, we need to support certain HTTP features:

- Caching headers for raising the cost-efficiency and user perceived performance,
- Content negotiation if multiple specifications should be supported,
- Cross Origin Resource Sharing headers for allowing the adoption in scripts on other domains.

Furthermore, we also need extra constraints on how we distribute and describe our dataset on the Web. The minimum set of items a client should be able to count on when retrieving a document is as follows:

- Every term must get an HTTP URI that results in a document containing a definition,
- There must be an RDF 1.1 serialization as a cross-standard way of understanding the domain model,
- There must be links to earlier versions of the data,
- There must be a description of the data license in the HTTP response.

In order to speed up some use cases where large analytical queries are needed, we can extend the basic paged collection of datasets with a multidimensional interface listing statistical summaries. This however goes beyond the Hydra vocabulary today [14].

5 PRELIMINARY OPEN DATA POLLING VS. WEBSOCKETS EXPERIMENT

Publishing live data on the Web can be done either through Publish/-Subscribe (pubsub) or Polling interfaces. Each of these publication strategies present different trade-offs regarding cost-efficiency of the server interface, response time latency, CPU usage and bandwidth consumption. Depending on the use case, there may be different constraints that could be fulfilled by either one of the publication strategies. Therefore, in order to have a clearer picture of how these strategies behave in different scenarios, we designed an experiment to measure server CPU usage and client-perceived latency for a Websockets-based pubsub interface and a HTTP-based polling interface, both exposing live data updates.

The live data stream used for the benchmarks consists of parking site availability observations as the ones used by our previously described proof of concept. The data is provided in RDF and serialized using the Trig format. Each observation is part of a named graph, annotated using the *http://www.w3.org/ns/prov#generatedAtTime* predicate to specify the moment the observation was performed.

5.1 Evaluation design

For the experiment we developed a set of configurable tools that comprise a complete scenario for publication and consumption of live data updates on the Web. Next we describe each of these tools:

LPDGenerator The Linked Parking Data Generator tool creates a dataset of parking availability observations. It is possible to configure the number of parking locations, the interval of time and the behaviour of the availability by defining what we call events, which help to mimic peak hours throughout the days.

Replay-Timeseries This tool takes a dataset of observations and starts to emit each of them at a predefined rate on the standard output. It is possible to configure the period on which the observations will be emitted, allowing to simulate faster or slower data streams.

Timeseries-Server This is a Node.js server, based on the Koa framework that receives updates through the standard input and exposes them through a HTTP interface where a client can request the last update. It also pushes the updates through a Websockets interface to every client previously subscribed. The server stores every update on a file-based paged collection, enabling access to historic data through a HTTP interface. It also provides metadata using the DCAT-AP specification, containing a description of the available streams, their access method (Websockets or HTTP) and their URL.

Nginx A Nginx instance is configured on top of the Timeseries-Server and used as a HTTP cache manager.

Timeseries-Client A Node.js application that can consume RDF streams either through HTTP polling or through a Websockets communication. It requests a DCAT-AP descriptor and proceeds to consume the described streams through their corresponding protocol. For HTTP polling the client uses the Max-Age value from the Cache-Control header to determine automatically when to request the next update. For a Websocket subscription it just awaits the next update from the server. In both scenarios the client measures the perceived latency of the data using the generatedAtTime value and the moment of reception. For this is necessary to synchronize the client's clock with the server, which is done by the method described in *http://www.mine-control.com/zack/timesync/timesync.html*.

The setup of the experiment is done using Docker containers and the docker-compose application to link all the containers together. In this paper we present some preliminary results that were obtained using a laptop with an Intel Core i5-7440HQ @ 2.8GHz x 4 and 16Gb of RAM for the server, and a laptop with an Intel Core i7 @ 3.6Ghz and 12Gb of RAM for the clients.

5.2 Inconclusive Results and Discussion

The results on the latency, illustrated in fig. 3, show no clear trend until 200 clients and are therefore inconclusive. During the test period, we also measure the CPU-time the server was consuming. The maximum spikes are visualized in fig. 4. Overall, the CPU consumption for both polling and pubsub remained low otherwise (<3%). If anything we can conclude from this test, is that we cannot conclude anything from an experiment up to 200 clients. For higher number of user agents, we will however need to amend our tests to work in a more distributed setting.

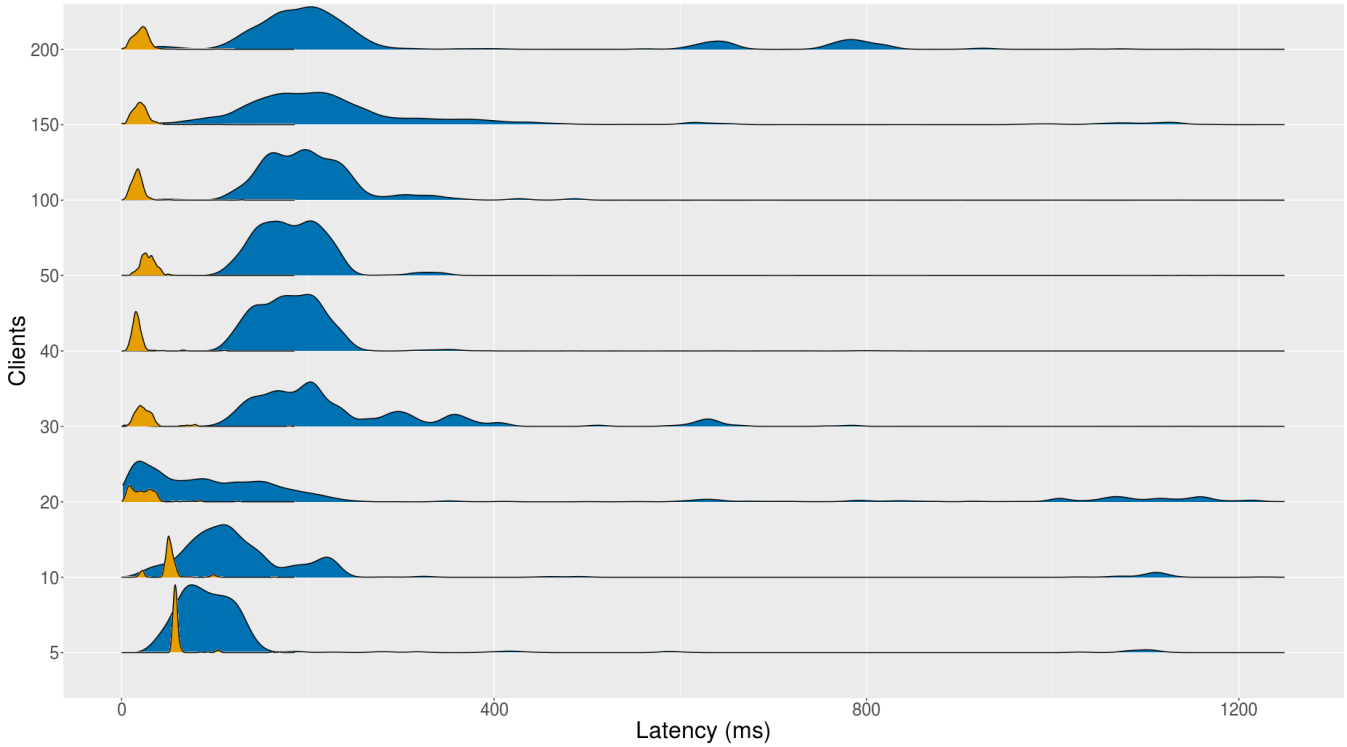

Figure 3: The Websockets (orange, left) approach uptil 200 clients shows no sign of slower latency or scalability issues, compared to the polling approach (blue, right).

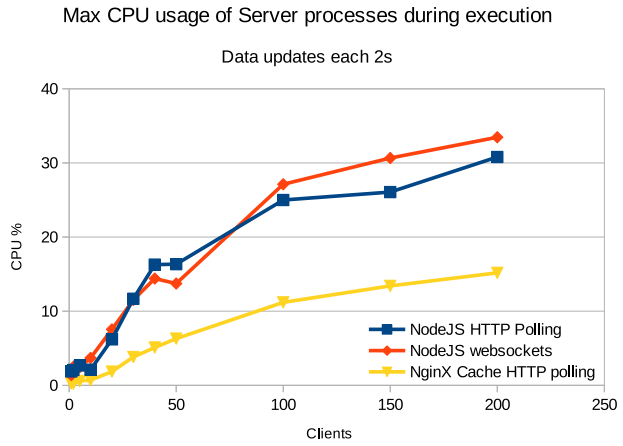

Figure 4: The maximum CPU% measured by dockerstats per process. For both HTTP polling and Websockets, there is no influence to be seen.

6 PROOF OF CONCEPT IMPLEMENTATION: LIVE OFF-STREET PARKING SITE AVAILABILITIES

In Ghent, a URI strategy is in place to create identifiers since 2016. The URI strategy defines a base URI at "https://stad.gent/id/". Using this strategy, the city introduced URIs for each parking site. We reused these identifiers to create a proof of concept at *https://datapiloten.be/parking*. As a proof of concept, we created a proxy for parking site availability across all of Flanders. We applied the above rules and have republished this dataset in a linked dataset(source code is available[5]).

A javascript NPM library was created (*https://www.npmjs.com/package/smartflanders-data-query*) to adopt this data in any kind of system. This library may aid a developer to write a program answering custom questions over these time series. This client automatically discovers certain building blocks, and can work with data published by others as well.

The approach was copied to the other centre cities and made available. Therefore we have a stream for each city (Kortrijk, Gent, Leuven and Sint-Niklaas), each one containing observations, every 30 seconds, regarding the parking availability of their public parking lots. Different streams can now be compared, without being centralized first, as illustrated in the screenshot in fig. 5. Also, fig. 5 shows access to parking availability historic data.

[5] *https://github.com/smartflanders*

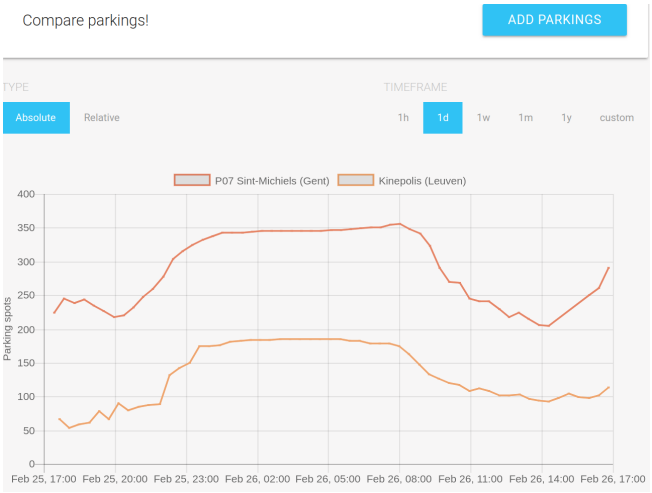

Figure 5: Screenshot of the demo web application to compare live and historic observations of parking availabilities at *https://smartflanders-poc.netlify.com*

7 CONCLUSION AND FUTURE WORK

In this paper, we identified key requirements for publishing live open data. These are currently the minimum requirements for the cities in Flanders. The Web – and the HTTP protocol more specifically – already contains the needed technological building blocks to publish live Open Data. However, there is no specification for live open data currently available telling all stakeholders within the Web ecosystem what building blocks to use. To that extent, in the region of Flanders, we are proposing eight minimal requirements to be added to an Open Data charter: (1) fragment your datasets and publish the documents over HTTP; (2) add caching headers to each document; (3) add hypermedia description using the Hydra vocabulary; (4) A web address (URI) per thing (i.e. everylike observation or sensor); (5) add a link to a machine readable open license in the document; (6) Add a Cross Origin Resource Sharing header enabling access from pages hosted on different origins; (7) in order to allow data portals to make your datasets discoverable, publish DCAT-AP metadata. When these technical principles are implemented, clients can benefit from these simple Web building blocks and start reusing this data automatically.

In "On a Web of Data Streams" [8], the authors put forward, in their first requirement[6], that the data should keep moving and should use a non-polling approach. Due to the nature of Open Data, and the fact the data becomes uncacheable in a push-based approach, we question whether this requirement holds true. After all, it is up to the data publisher to make a trade-off: how important is not working with stale data vs. what budget is there for hosting the data? We performed a first test, from which we conclude that we will need a more advanced experiment: up to 200 clients, there is no trend to be seen, and we cannot draw any conclusions.

Due to the constraints of HTTP, caching is not possible for subsecond streams, neither is optimizing polling time, as also the Age header should be in seconds. In future work we will conduct more experiments on how fast data should update in order for HTTP polling to become suboptimal, regardless of the spec.

In the use case of parking site availabilities for the centre cities in Flanders, we conclude that the live data using HTTP documents instead of a pull protocol is a good choice: implementation is kept simple and the data is easily described with standard Linked Data and HTTP technologies.

Multidimensional Interfaces [14] were introduced for fragmenting data with an order generically, and publishing these fragments in an interface-level index. These interfaces can make multidimensional ordinal data such as timeseries or geospatial data automatically discoverable and consumable by clients using hypermedia controls. In future work, we will use this interface to provide statistical summaries over longer periods of time, and thus allow agents to speed up certain analytical queries over the historic observations. Furthermore, metadata can this way make time series automatically discoverable, whether push or poll based.

[6] *https://dellaglio.github.io/webstreams/index.html#sec:scenario-req-1*

ACKNOWLEDGEMENTS

This work was funded by the Smart Flanders program (*https://smart.flanders.be*). Thank you to communication scientists Nils Walravens and Mathias Van Compernolle for the endless inspirational collaboration on how to make Open Data work organizationally. A special thank you to the 13 centre cities and the VGC Brussels for their cooperation.

REFERENCES

[1] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-sparql: Sparql for continuous querying. In *Proceedings of the 18th international conference on World wide web*, pages 1061–1062. ACM, 2009.
[2] D. F. Barbieri and E. Della Valle. A proposal for publishing data streams as linked data-a position paper. 2010.
[3] R. Buyle, P. Colpaert, M. Van Compernolle, P. Mechant, V. Volders, R. Verborgh, and E. Mannens. Local council decisions as linked data: a proof of concept. In *Proceedings of the 15 th International Semantic Web Conference. Kobe, Japan, October 17*, volume 21, 2016.
[4] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th international conference on World Wide Web*, pages 613–622. ACM, 2005.
[5] P. Colpaert. *Publishing Transport Data for Maximum Reuse.* PhD thesis, Ghent University, 2017.
[6] P. Colpaert, R. Verborgh, and E. Mannens. Public transit route planning through lightweight linked data interfaces. In *International Conference on Web Engineering*, pages 403–411. Springer, 2017.
[7] D. Dell'Aglio, E. Della Valle, J.-P. Calbimonte, and O. Corcho. Rsp-ql semantics: a unifying query model to explain heterogeneity of rdf stream processing systems. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(4):17–44, 2014.
[8] D. Dell'Aglio, D. Le Phuoc, A. Le-Tuan, M. I. Ali, and J.-P. Calbimonte. On a web of data streams. 2017.
[9] R. T. Fielding, R. N. Taylor, J. Erenkrantz, M. M. Gorlick, W. E. J., R. Khare, and R. Oreizy. Reflections on the rest architectural style and "principled design of the modern web architecture". In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 4–11, 2017.
[10] S. Harris, A. Seaborne, and E. Prud'hommeaux. Sparql 1.1 query language. *W3C recommendation*, 21(10), 2013.
[11] D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *International Semantic Web Conference*, pages 370–388. Springer, 2011.
[12] A. Mauri, J.-P. Calbimonte, M. Balduini, E. Della Valle, K. Aberer, et al. Where are the rdf streams?: Deploying rdf streams on the web of data with triplewave. In *14th International Semantic Web Conference ISWC*, number EPFL-POSTER-212733, 2015.
[13] M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 requirements of real-time stream processing. *SIGMOD Rec.*, 34(4):42–47, Dec. 2005.
[14] R. Taelman, P. Colpaert, R. Verborgh, P. Colpaert, and E. Mannens. Multidimensional interfaces for selecting data within ordinal ranges. In *Proceedings of the*

7th International Workshop on Consuming Linked Data, Oct. 2016.

[15] R. Taelman, R. Verborgh, P. Colpaert, and E. Mannens. Continuous client-side query evaluation over dynamic linked data. In *The Semantic Web: ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 – June 2, 2016, Revised Selected Papers*, pages 273–289. Springer International Publishing, May 2016.

[16] P.-Y. Vandenbussche, G. Atemezing, M. Poveda-Villalón, and B. Vatant. Linked open vocabularies (lov): a gateway to reusable semantic vocabularies on the web. *Semantic Web*, 8(3):437–452, 2017.

[17] G. Vandewiele, P. Colpaert, O. Janssens, J. Van Herwegen, R. Verborgh, E. Mannens, F. Ongenae, and F. De Turck. Predicting train occupancies based on query logs and external data sources. In *Proceedings of the 7th International Workshop on Location and the Web*, Apr. 2017.

[18] R. Verborgh. Your JSON is not my JSON – a case for more fine-grained content negotiation. In *Proceedings of the Workshop on Smart Descriptions & Smarter Vocabularies*, Nov. 2016.

[19] R. Verborgh, M. V. Sande, O. Hartig, J. V. Herwegen, L. D. Vocht, B. D. Meester, G. Haesendonck, and P. Colpaert. Triple pattern fragments: A low-cost knowledge graph interface for the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37-38:184 – 206, 2016.