Providing Reliable Access to Real-Time and Historic Public Transport Data Using Linked Connections

Julián Rojas[1], David Chaves[2], Pieter Colpaert[1], Ruben Verborgh[1], Erik Mannens[1]

[1]IDLab, Department of Electronics and Information Systems, Ghent University - imec
julianandres.rojasmelendez@ugent.be

[2]Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
dchaves@fi.upm.es

Abstract. Using Linked Data based approaches, public transport companies are able to share their time tables and its updates in an affordable way while allow- ing user agents to perform multimodal route planning algorithms. Providing time table updates, usually published as data streams, means that data is being constantly modified and in the presence of large analytical queries, results might be affected due to the changing data. In this demo we introduce a mecha- nism to tackle this problem by guaranteeing that a user agent will always re- ceive version based responses, therefore ensuring data consistency. Such mech- anism also enables access to historical data that could be used for deep analysis of transport systems. However, how this data shall be archived, in order to keep this approach scalable and inexpensive is still a matter of study. In a demonstra- tor, we published and query data from the Belgium national train system (SNCB) and Madrid Regional Transport Consortium (CRTM). This paper rep- resents the first step towards establishing an affordable framework to publish reliable transport data.

Keywords: Semantic Web, Open Linked Data, Linked Data Fragments, Linked Connections, public transit, reliability, content negotiation.

1 Introduction

Open Transport Data publishers, such as a public transport company, can share their time tables and its updates for maximum reuse by studying the trade-offs introduced by the Linked Data Fragments axis [1]. Choosing a use-case specific fragmentation strategy, can result in a cost-efficient data publishing interface while still being able to evaluate queries. The Linked Connections (LC) framework [2] enables third parties to develop route planning algorithms that evaluate queries over different data sources, taking into account multiple modes of transport (e.g., train, bus or tram). To achieve this, the server interface only exposes a paged collection of an ordered public transit connections list.

By updating these fragments, updates on the planned schedules can be handled through the LC framework as well. However, this means that data is constantly changing and if there is a large analytical query, its response might be affected in terms of reliability and

data consistency. When a client would thus issue a large query, the data processed at the start of the query would be correct but might change at the end of the query evaluation. This could turn query results unreliable or even inconsistent.

To tackle this issue, this demo introduces a mechanism provide time-based versions of data which ensures that a client will always get a true result for a given moment. Furthermore, having time-based versions of data enables access to historic data, allowing deep behavioral analysis of transport systems. We first describe the background, we then describe the implementation details of our approach and finally we present the set-up of our demonstrator.

2 Background

In related work, the *General Transit Feed Specification* (GTFS)[1] defines a set of rules to describe a transport system schedules using a CSV format. To handle data updates, GTFS-RT was introduced. In our approach we use GTFS and GFTS-RT feeds provided by public transport agencies as input data for our system.

A connection in the LC framework is defined as hop from one transit stop to another, indicating location and time of departure, and location and time of arrival [2]. Moreover, to provide context to HTTP user agents and link the terms and identifiers of GTFS and Connections to the Linked Open Data cloud, the Linked GTFS[2] and LC[3] ontologies were introduced.

The Memento Framework provides a mechanism to achieve a tighter integration between the present and past Web [3]. It introduces the *datetime* dimension for content-negotiation between a server and a client, allowing to access previous versions of a resource. The latest version of a resource is defined as the *original resource* URI-R. Previous versions of URI-R are defined as *Mementos* URI-Mi (i = *1...n*). Memento defines different patterns of implementation, on this approach we use *Pattern 1*[4] for simplicity, but other patterns shall be studied in future work such as the one described in [4].

3 Implementation

The implementation for this demo was built on top of Node.JS and its source code is available on GitHub[5]. Figure 1 portraits a modular architecture of the system where main modules and their relations are presented.

[1] https://developers.google.com/transit/gtfs/

[2] http://vocab.gtfs.org/terms

[3] http://semweb.datasciencelab.be/ns/linkedconnections#

[4] Memento is formally specified in RFC 7089 available at https://tools.ietf.org/rfc/rfc7089.txt

[5] https://github.com/julianrojas87/linked-connections-server

The *Fetching Module*, as its name implies, is in charge of fetching GTFS and GTFS-RT feeds published by public transport agencies and converting them to LC sorted by departure time.

The *Archiving Module* takes the LC of transport agencies and stores a time-based fragmentation of the data as paged files. Updates obtained from GTFS-RT feeds provide data concerning specific connections. Every time table update is annotated with a timestamp of the moment it was issued. Such annotation enables datetime based content-negotiation, as specified by Memento, to provide the state of connections at a certain given time.

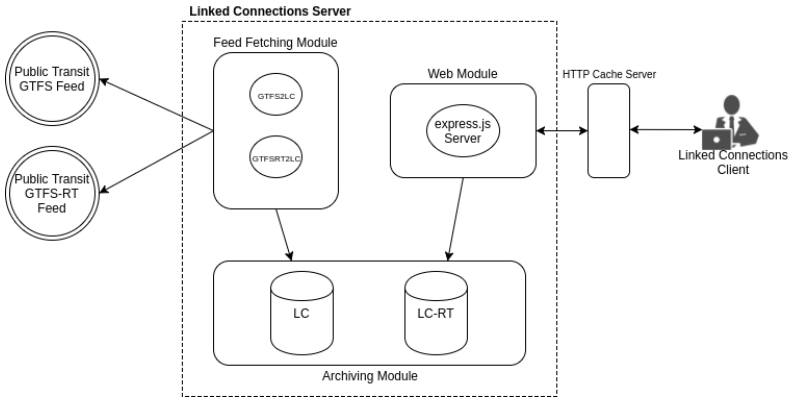

Fig. 1. Modular architecture of the Linked Connections Server

The *Web Module* is a Express.js based web server which handles HTTP queries for connections with a given departure time. Once a query is received, the server retrieves the corresponding LC fragment for the specified departure time, then it retrieves the available updates at the time the query was issued for every connection in the fragment. This ensures that the client will always receive the state of all connections at the moment it issued its query, regardless of whether at the end of it, new data is already available. Finally a web representation of the LC is created and serialized in JSONLD format, containing links to other resources such as next and previous LC fragments using *hydra:nextPage*[6] and *hydra:previousPage* links. Furthermore, by using the Accept-Datetime header, clients are able to query for connections status at a certain given time enabling access to historic data. An HTTP Cache Server may be configured on top of the LC Server in order to improve performance on recurrent queries.

4 Demonstrator

A demonstrator performing long analytical queries, in need of data consistency, can be viewed at *http://tripscore.eu*[7]. The application aims at presenting to the user a list

[6] *hydra:* is a prefix that can be expanded to https://www.w3.org/ns/hydra/core#

[7] Source code available at https://github.com/oSoc17/oasis-frontend/

of trip alternatives for a specific route, with a calculated quality score based on predefined criteria such as maximum number of hops, acceptable delay times, minimum transfer times and delay consistency[8]. The user can enter his/her preferences and query for a route he/she frequently uses. The application will start querying for connections and will begin to show alternatives to the user as it keeps processing the data. At the same time it will show the amount of HTTP requests and responses handled and the number of LC analyzed.

In order to calculate the quality score of a trip and its delay consistency, the application will query data spanning the last 5 weeks if available, demonstrating an use case for historic data. As mentioned before, this demo publishes and allows to query data from the Belgian railway company (SNCB) and Madrid transport system (CRTM).

5 Conclusion

In this demo we used the LC framework to publish and query public transport data in an affordable way, and extended its capabilities by introducing a time based versioning mechanism that ensures reliable and consistent responses to large analytical queries. Furthermore, our approach also enables access to historical data that could be used to perform different type of analysis as showcased by our demo which calculates delay consistency over time for specific trips.

This work represents a step forward towards a reliable and inexpensive mechanism to publish public transport data that allows multimodal route planning and analysis of historical data. However several issues still remain unattended such as studying the best approach to store LC in a scalable way, finding the best way to fragment transport data in order to improve performance of route planners and how the system may discover new transport datasets reliably.

References

1. Verborgh, R., Hartig, O., Meester, B.D., Haesendonck, G., Vocht, L.D., Sande, M.V., Cyganiak, R., Colpaert, P., Mannens, E., Walle, R.V.D.: Querying Datasets on the Web with High Availability. The Semantic Web – ISWC 2014 Lecture Notes in Computer Science. 180–196 (2014).
2. Colpaert, P., Llaves, A., Verborgh, R., Corcho, O., Mannens, E., Van de Walle, R.: Intermodal public transit routing using Linked Connections. In: Proceedings of the 14th International Semantic Web Conference: Posters and Demos (2015).
3. Van de Sompel, H., Sanderson, R., Nelson, M.L., Balakireva, L., Shankar, H., Ainsworth, S.: An HTTP-based Versioning Mechanism for Linked Data. CoRR abs/1003.3661 (2010).
4. Sande, M.V., Colpaert, P., Nies, T.D., Mannens, E., Walle, R.V.D.: Publish data as time consistent web API with provenance. Proceedings of the 23rd International Conference on World Wide Web - WWW '14 Companion. (2014).

[8]Refers to the behavior regarding delays for a specific connection during a given period of time.