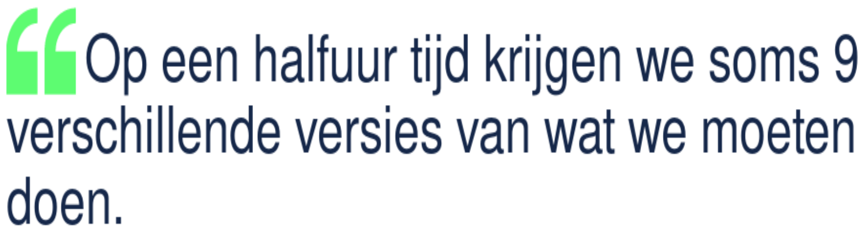
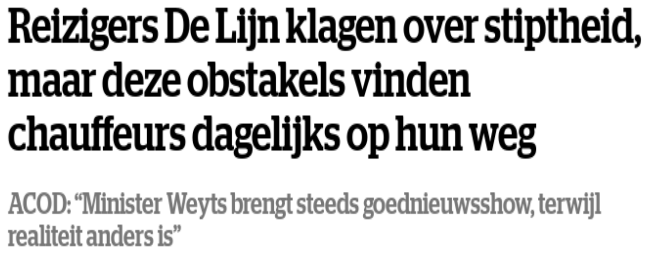
Publishing real time Open Data

Are you sure you’re going to get there?



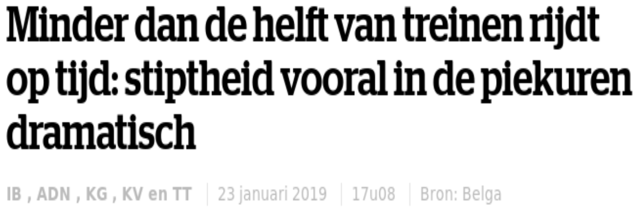
Op een halfuur tijd krijgen we soms 9 verschillende versies van wat we moeten doen.

Treinbegeleider



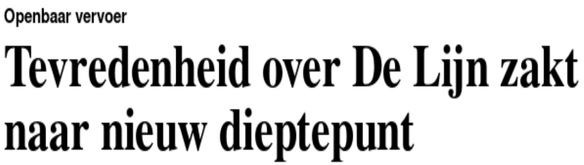
Reizigers De Lijn klagen over stiptheid, maar deze obstakels vinden chauffeurs dagelijks op hun weg

ACOD: “Minister Weyts brengt steeds goednieuwsshow, terwijl realiteit anders is”



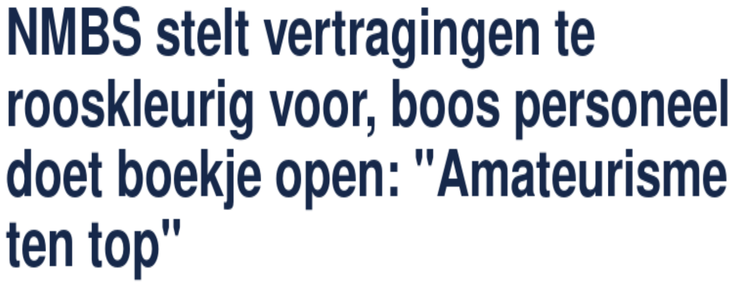
Minder dan de helft van treinen rijdt op tijd: stiptheid vooral in de piekuren dramatisch

IB , ADN , KG , KV en TT | 23 januari 2019 | 17u08 | Bron: Belga

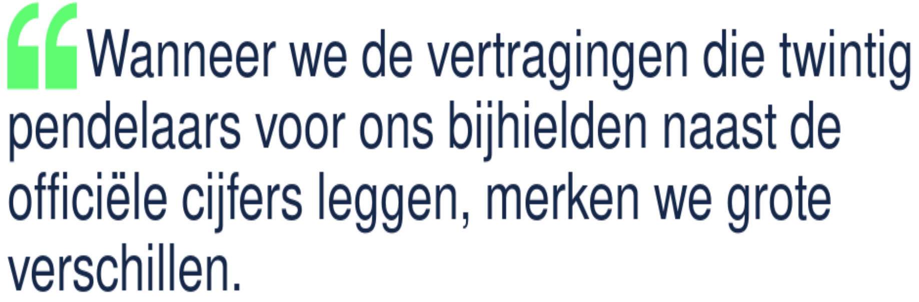


Openbaar vervoer

Tevredenheid over De Lijn zakt naar nieuw dieptepunt



NMBS stelt vertragingen te rooskleurig voor, boos personeel doet boekje open: "Amateurisme ten top"



Wanneer we de vertragingen die twintig pendelaars voor ons bijhielden naast de officiële cijfers leggen, merken we grote verschillen.

Research questions

- How can we keep a cache for route planner up to date in a cost-efficient way for both Open Data publishers and consumers?
- How can we consume real time Open Data updates in route planner algorithms in an efficient way?

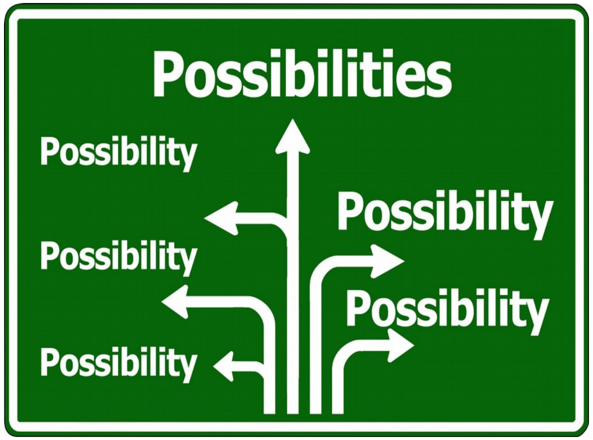
- **Specifications**
- Implementation
- Experiments
- Conclusion

Goals

Informing commuters in real time

Alternatives based on real time data

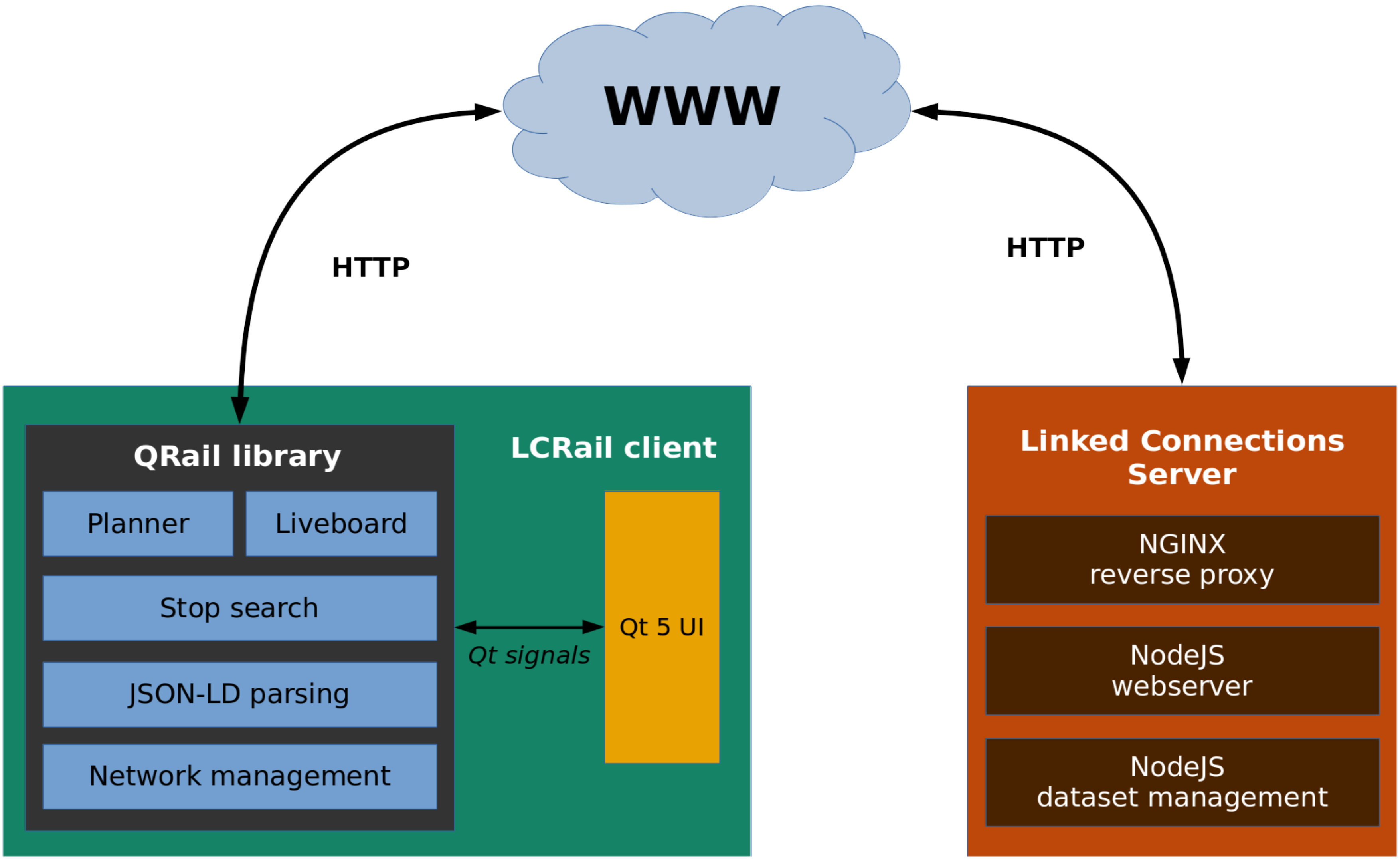
Cost-efficient for publishers and users



Linked Connections

- Specifications
- **Implementation**
- Experiments
- Conclusion

Architecture



Our improvements to Linked Connections

1) **Gtfsrt2lc**: Delays and cancellations added
2) **Realtime resource**: Added to the Linked Connections Server
3) **Connection Scan Algorithm**: rerouting support with real time updates
4) **Liveboard algorithm**: departing vehicles in a station with real time updates

Adding a real time resource to Linked Connections

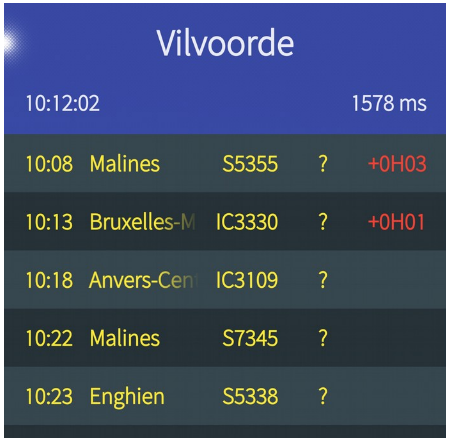
1. Publishing of the real time data

Linked Connections Server

2. Efficiently using the real time data

CSA & Liveboard algorithms in QRail

3. Informing the user



Vilvoorde

10:12:02 1578 ms

10:08	Malines	S5355	?	+0H03
10:13	Bruxelles-M	IC3330	?	+0H01
10:18	Anvers-Cen	IC3109	?	
10:22	Malines	S7345	?	
10:23	Enghien	S5338	?	

LCRail client

Format & transport protocol

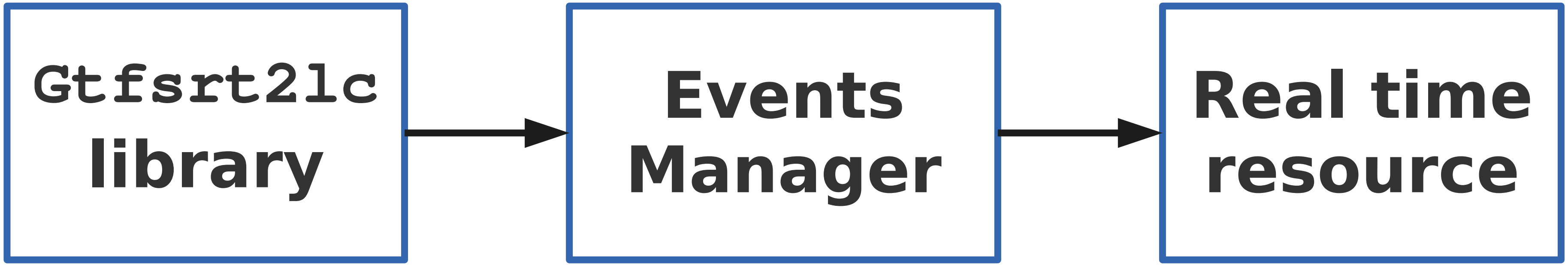
HTTP	JSON
Supported by every Web client	Supported by every Web client
Readable by humans & machines	Readable by humans & machines
HTTP caching	Lightweight data format

`Gtfsrt2lc` library

- Completely rewrote the algorithm to convert the GTFS-RT data to Linked Connections
- Previously, some delays were skipped
- Added support for cancelled vehicles

PICTURE

Events Manager



Gtfsrt2lc library

Events Manager

Real time resource

Generates all events from the last X minutes

- Publishes only new events
- SOSA ontology

- HTTP polling
- SSE

Adding a real time resource to Linked Connections

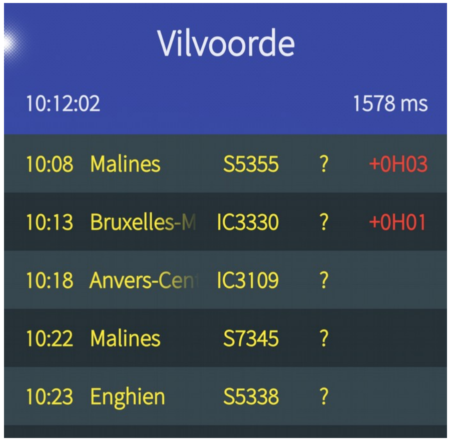
1. Publishing of the real time data

Linked Connections Server

2. Efficiently using the real time data

CSA & Liveboard algorithms in QRail

3. Informing the user



Vilvoorde

10:12:02 1578 ms

10:08	Malines	S5355	?	+0H03
10:13	Bruxelles-M	IC3330	?	+0H01
10:18	Anvers-Cen	IC3109	?	
10:22	Malines	S7345	?	
10:23	Enghien	S5338	?	

LCRail client

Connection Scan Algorithm routing in QRail

Routing updates in QRail

Liveboard algorithm with updates in QRail

Adding a real time resource to Linked Connections

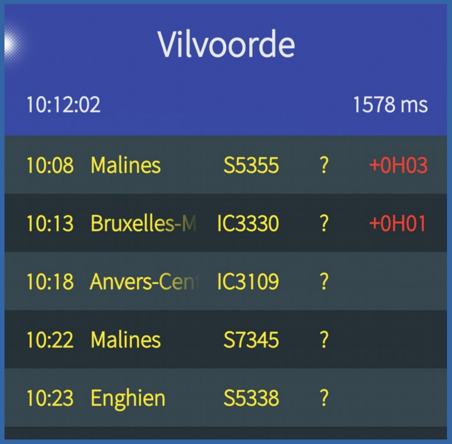
1. Publishing of the real time data

Linked Connections Server

2. Efficiently using the real time data

CSA & Liveboard algorithms in QRail

3. Informing the user



Vilvoorde

10:12:02 1578 ms

10:08	Malines	S5355	?	+0H03
10:13	Bruxelles-M	IC3330	?	+0H01
10:18	Anvers-Cent	IC3109	?	
10:22	Malines	S7345	?	
10:23	Enghien	S5338	?	

LCRail client

LCRail: informing the user

- Rerouting

- Liveboard

- Specifications
- Implementation
- **Experiments**
- Conclusion

Test environment

- **Pushing vs polling experiment**: Digital Ocean server & Virtual Wall clients
- **Processing of updates experiment**: Digital Ocean server & Sailfish OS clients
- **Conditions**: Each experiment ran for 30 minutes with a clean cache

DigitalOcean

SAILFISH OS

Pushing vs polling: CPU & RAM usage

- CPU + RAM server HTTP

- CPU + RAM server SSE

Processing of updates: CPU & RAM usage

- CPU

- RAM

Processing of updates: bandwidth usage

- Sent

- Received

Processing of updates: User informed time

- CSA

- Liveboard

Demo: Connection Scan Algorithm in LCRail

Demo: Liveboard in LCRail

- Specifications
- Implementation
- Experiments
- **Conclusion**

How can we keep a cache for route planners up to date in a cost-efficient way for both Open Data publishers and consumers?

1) **Pushing**: Let the server push the data to the clients with SSE
2) **Real time resource**: Less bandwidth and processing power is needed
3) **Updates only**: Only process the changes to the data

How can we consume real time Open Data updates in route planner algorithms in an efficient way?

1) **Real time data processing**: Extra logic is needed
2) **Updates only**: Processing only the changes to the data, reduces the CPU and bandwidth usage
3) **Faster computation**: The user is informed faster about the changes to his or her trip

Thank you for your attention!

Are there any questions?