Paper ID #

(User friendly) Integration of on-demand fleets in public transport

Anke Sauerländer-Biebl[1*], Eric Neidhardt[2]

1. DLR, Institute of Transportation Systems, Germany,
anke.sauerlaender-biebl@dlr.de
2.. DLR, Institute of Transportation Systems, Germany,
eric.neidhardt@dlr.de

Abstract

On-demand driven public transport as extension for the traditional public transport is currently a popular topic and tested and implemented worldwide with different characteristics. The provider of such on-demand fleets are not necessarily the same company as the surrounding public transport provider. This makes a seamless use for the customers difficult: E. g. they need an additional app to check the availability and to book a ride. Also, the connection to and from the line based surrounding transport is often not ensured. This paper describes a platform approach to connect several on-demand fleets with the main public transport provider of a region. It was firstly tested in a rural area of Mecklenburg and it is also actively used in the project RealLabHH, subproject "Autonomous public driving" where an autonomous driving shuttle fleet with operation area in Hamburg-Bergedorf is connected to public transport.

Keywords: Intermodal Public Transport Platform, On-demand Fleets

Introduction

The demand of deploying on-demand traffic

On-demand traffic in public transport is of interest for areas where there is no connection to public transport at all or line and timetable-based traffic is not efficient enough (for provider and/or passengers). The motivation in urban areas are mostly different than the ones in rural areas. In rural areas you often find "self-organized" on-demand transport. They are not run by a traditional public transport provider but by civil non-profit organizations. Their aim is to bring people to the public transport, to the next city or to enable connections between villages. In urban and suburban areas on-demand traffic is often used to connect residential areas that are not (yet) connected to public transport. This on-demand busses are usually run by the local public transport provider themselves or subcontractors.

Digital integration

In all cases mentioned above the on-demand vehicle has to be booked or ordered. This is usually done by

offering a telephone number that the customers have to call. Modern systems that organize area-based on-demand fleets provide mobile apps for requesting and booking the transport. These apps are usually standalone solutions and not integrated in the digital apps of the local transport associations. So, the customers must be informed that they need a special app to get connections in an area served by an on-demand fleet.

The main problem of most public transport connection information systems is that they do not expect a feedback of the potential customer. Maximum comfort is to offer a link into the third-party app to do a reservation or booking or to display the phone number for reservation requests. The actual app does not know if the user books such an on-demand offer or not. Also, the app does not know if there is still capacity in the on-demand vehicles. In summary we can say that an interactive connection information system is necessary that reacts if users choose specific connections. The solution of the platform, described in this paper, is described in the sub section for the ConnectionService.

A further aspect is the payment of tickets. This paper will not handle the payment aspect. It is assumed that the on-demand traffic is integrated in the tariff of the public transportation association or that the ticket can be paid on the vehicle.

The "HubChain"-Platform – An approach

The DLR Institute of Transportation Systems developed a platform to combine on-demand fleets with surrounding public transport. The aim is to have one system for all public transport offers of a region. If a trip or parts of a trip need a reservation this is also done via the platform. This platform was developed in the project HubChain to manage on-demand fleets in Mecklenburg-Vorpommern in a rural area around the small town Röbel and to ensure the transition to the local line-based bus system. So, the system includes its own on-demand-fleet-management, designed to handle several fleets, but can also connect third-party solutions if they use defined APIs. The internal on-demand disposition is an area-based disposition with connection to defined "hubs" for transition to line based transport.

The platform is a backend system that offers web interfaces for

1. customer apps (for connection requests and information and to do and manage reservations)

as well as

2. a web interface to request the dispatched routes of the internal on-demand disposition so that an app for drivers or in a control center can show the planned routes and stops.

The name HubChain should accent the connection of different means of transportation between "hubs". A typical chain can be for example:

- City-bus → **Hub1** (City station) → Regional train → **Hub2** (Regional train station) → On-demand bus

or

- On-demand shuttle → **Hub1** (Bus station) → Regional bus (line) → **Hub2** (Bus terminal in next city)

The platform is designed to handle a region/district. Usually in a region or district there is one public

transport provider (or a public transport association) and several distributed on-demand fleets.

What makes it so difficult to integrate on-demand traffic in the conventional connection information? As mentioned above it is the need for interactive communication when bookable transport is involved. In general, there are three kinds of action that are necessary, independent if an on-demand trip or a transport with shared vehicles (e.g. bike sharing) or something similar is involved. Assuming the connection information system knows about these mobility offers, their operation area, and availability, then there are three actions that must be provided in the connection information app:

1. **Confirm** or do a booking
2. **(Re-)read** the booking data from the third-party system
3. **Cancel** a booking

External public transport provider have to register at the HubChain-platform. Because the platform was developed in Germany the two most used API-formats HAFAS or TRIAS are accepted for connection requests and responses.

The following chapter explains the components of the HubChain-platform.

Components of the platform

The operating system of the platform is a Linux Ubuntu distribution. As additional software ActiveMQ is installed for communication between the internal services. An Apache-Tomcat combination is used as webserver and the database is a PostGis database. All this software is free and can be used without paid license. The platform contains several internal Java services and modules that are able to receive and send messages among each other. A service is a module that starts working when it receives a request or a message of another service or module. Other modules are processing data as “batch” controlled by a time schedule and save their output in the database.

Figure X shows the components of the HubChain platform. They will be explained in the following text.

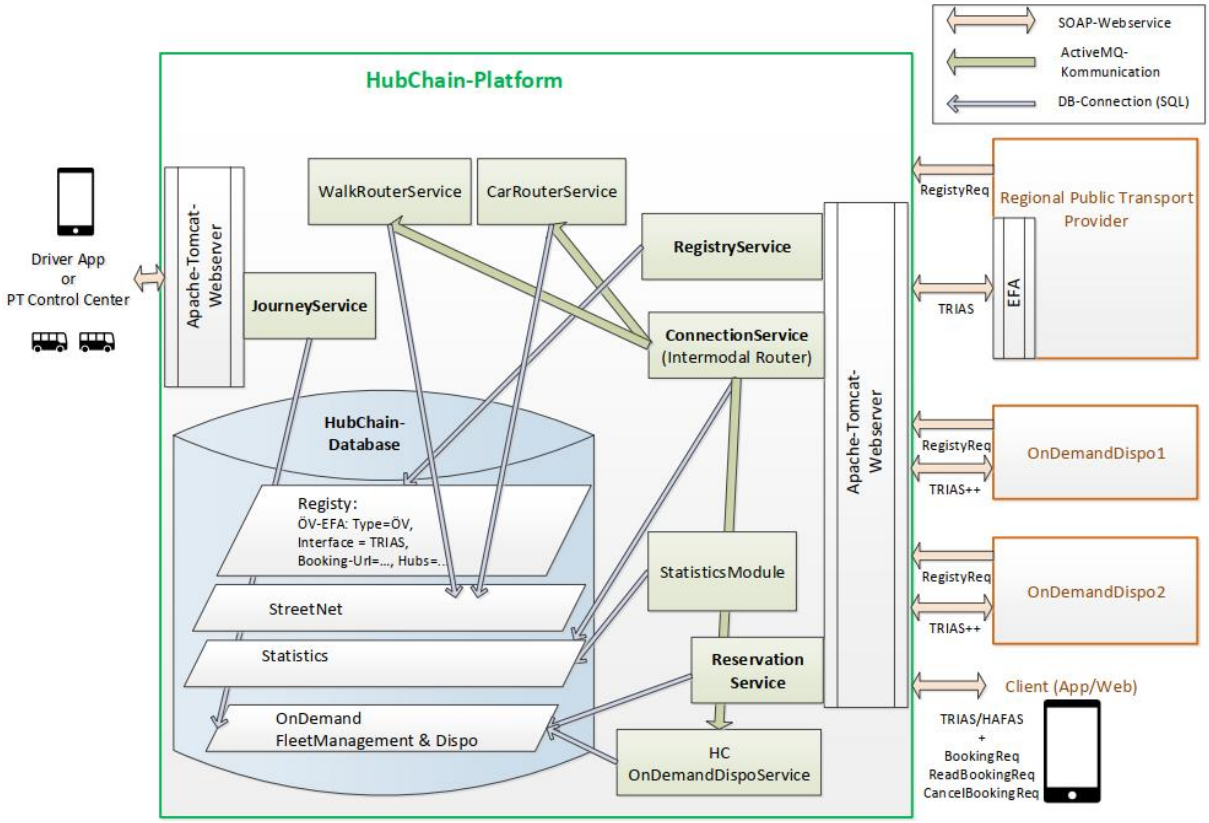

Figure 1: Components of the HubChain platform

Database

The intermodal platform includes its own database. It is necessary for storing the information about the responsibility and description of the connected external services as well as saving data the internal services work on and produce. The platform uses a PostGis database because geographical data plays an important role and is needed be the services. Of course, other database systems with spatial extension also runs. Especially a street network of the focused region or parts of it is needed. The database server includes

- a database for the registry entries of external public transport systems
- a database for a routable street net (needed e.g. for finding food path, …)
- a database for statistics and evaluation
- a database for the internal on-demand fleet management system (if required)

The databases run in one server and are linked to exchange information.

Services and Modules

RegistrationService:

The registration service is used by external dispatched on-demand fleets and conventional public transport provider to registrate their service at the platform. When having successfully registrated the new service must still be manually set to “active” to avoid misuse. The minimum set of data that is

needed is:

type:	Set of enumerations like ON_DEMAND_BUS, BIKE_SHARING, …
name:	Name of the service
valid from:	Date from which the service is usable
valid until:	Date until the service is usable
interface format:	Format of the interface: TRIAS, HAFAS, DLR
operation area:	Geographical description of the operating area
service ip:	IP or host name
service port:	port that is to use in combination with the is/hostname
service id:	A identification string of the service

To define favorite transition positions (hubs) an optional list with stops/station can be added. They are described by their geographical position, name, operator-id and operator name. This transition locations are used when the ConnectionService is combining different service providers. Also, the connected service can influence in that way the preferred transition stops or stations.

Multi- and intermodal ConnectionService:

This is the core service of the platform. It receives connection requests. It needs start and destination coordinates, the desired latest arrival or earliest departure time and some additional service options. To determine suitable connections the service can request internal routers (for walk, bicycle or car sequences), on internal on-demand dispatched fleets, on external public transport APIs and on external on-demand fleets. The router tries to combine the connected provider if necessary.

First step of the router strategy is to request the conventional public transport. As next step it checks if the start or end location belongs to an area that is served by an on-demand fleet. If this is the case and both locations are inside on operation area this on-demand service is requested for a connection. If only start or destination location belongs to an on-demand served area or they belong to different services, a suitable combination has to be found. The intermodal router tries to find the best transition stops/stations and divides the request into appropriate sub-requests to the different providers. Its strategy is visualized in Figure 2: Strategy of intermodal router. If a start or destination location belongs to several operation areas of each of the systems is requested. Assuming the start location belongs to the operation area of the conventional public transport and also in the operation area of an on-demand fleet. Then two journey proposal will be produced (if available): The first one is a journey only with the conventional public transport and the second one is a journey who’s first leg is an on-demand transport to a defined transition stop and a second one with the conventional line-based transport. The same would apply to the end of a journey if the destination belongs to two operation areas.

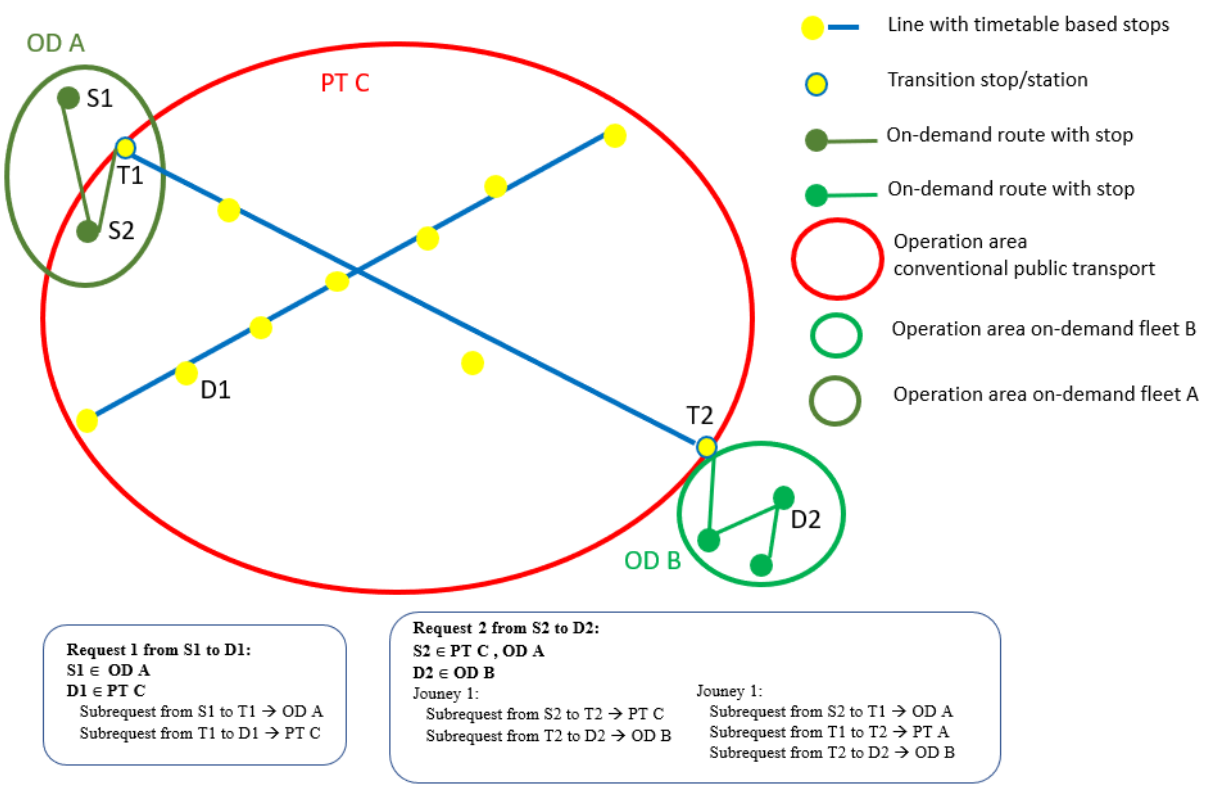

Figure 2: Strategy of intermodal router

If several journeys for a request are calculated, all of them are returned to the calling client app, except a ratio of travel time and distance is exceeded.

When returning the calculated connection proposals, independent if using the DLR-format, TRIAS or HAFAS, a new additional element is added to the connection segment if it is an on-demand segment. The counterpart to what is named “segment” in this paper is for

- HAFAS format: A connection section “ConSection”
- TRIAS format: A trip leg “TripLeg”

These elements are extended with the “DrtPassengerInfoElement”. This element contains all info about the trip and the possibilities to conform or cancel the reservation for it. This element is also suitable to store information for area-based on-demand-trips that have variable get-on and get-off times within a given time window and who’s final times are fixed X minutes before the vehicle journey starts or before earliest departure. Here is the definition:

```
DrtPassengerElement:
Long passengerRequestId: Unique ID of the passenger’s request
Long passengerId: Optional ID for the passenger
GeoPosition originPosition: Start position
GeoPosition destinationPosition: End position
int numberOfPassengers: Number of passengers
Date originallyPlannedDepTime: The first calculated departure time
Date originallyPlannedArrTime: The first calculated arrival time
```

```
Date currentlyPlannedDepTime: The currently planned departure time
Date currentlyPlannedArrTime: The currently planned arrival time
Date delayedDepTime: Delayed departure time, if real-time info available
Date delayedArrTime: Delayed arrival time, if real-time info is available
Location currPlannedOrigLocation: Location object for start location (optional)
Location currPlannedDestLocation: Location object for destination (optional)
Date earliestDepartureTime: Earliest departure time if given in request
Date latestArrivalTime: Latest arrival time if givne in request
Date requestTimestamp: Timestamp of passenger's request
Long journeyId: Unique ID of a vehicle journey if dispatched by HubChain-Dispo
Integer state: State of the reservation: Preliminary, Confirmed, Final, Cancel,
String bookingNumber: Reserveration number
String vehicleInformation: Additional information of the vehicle, e.g. name
String contactInformation: Optional contact information of the passenger
String informationText: Additional information like "Bus has to be booked"
String urlConfirm: Link to confirm the reservation
String urlGetBookingData: Link to read actual data of reservation.
String urlCancel: Link to cancel the reservation
```

The last three URL fields contains a "link" (URL with complete message body) to confirm, (re-)read and cancel a reservation/booking. This is important for external apps to manage a customer's reservation that he has done via this app. With this interaction of the connection information app and the reservation system of on-demand or any other bookable traffic is possible.

Walk&CarRouterServices

This routing services are used by the ConnectionService to add missing segments to connections requested from external fleets. Examples are missing walk segments to the first or from the last stop/station or if no public transport route could be found, a bike or car route can be returned.

(HubChain-)OnDemandDispositionService:

This service is an internal, optional Service that can manage several on-demand fleets, including their dispatching. It is based on two database schemas: One for the description of the fleets and an other one for the disposition data. The fleet schema contains all static data of fleets like, name, operation area and times, constraints for the disposition, …. The disposition schema contains the "living" data, produces by the disposition like passenger requests, vehicle routes, passenger bookings, planned stops, ….

The optimization algorithm is a modified and extended version of the ant colony approach. More details about it can be found in [1].

ReservationService

The ReservationService is needed if the platform internal OnDemandDispositionService is used. The

ReservationService handles the reservations for the passenger's on-demand trips. The service has methods to confirm a reservation, to read a specific reservation and to cancel a reservation. It is deployed as a web service so that external clients can manage bookings/reservations of a customer.

JourneyService
The JourneyService is also only needed if the platform internal OnDemandDispositionService is used. It is a webservice that delivers the calculated routes, including stops, to external clients. Apps for drivers of on-demand shuttles or control center application of the fleet provider needs this information to control their vehicles.

Conclusion and outlook

Modern on-demand and sharing mobility offers are developed by different companies with different background. Some have been or are start-ups that do not necessarily adopt their interfaces to common formats used in conventional public transport. For customers there is the urgent need to have mobility services from one source. They would like to request connections for a region that include all mobility offers. First step is one information and booking app, the second step is a comprehensive single-sign-on with integrated payment. First approaches can be found, but usually only a selection of mobility providers are considered. If an interface format to agree on is found, it is conceivable that not private companies offer Mobility-as-a-Service apps but instead the region or city administrations to ensure competitive equality in favor of the passengers.

The HubChain-platform was developed in the project HubChain that was financed by the BMVI :

HUB CHAIN

Bundesministerium für Wirtschaft und Energie

and it is further developed and adopted to autonomous drivng on-demand fleets in the running project RealLabHH, subproject 4 "Autonoumous public driving" (https://reallab-hamburg.de/projekte/autonomes-fahren/) that is financed by the BMVI:

RealLab Hamburg

Gefördert durch:

Bundesministerium für Verkehr und digitale Infrastruktur

aufgrund eines Beschlusses des Deutschen Bundestages

References

1. Tang, Q., Neidhardt, E. (2019). Simulation-based method of a dynamical on-demand transportation problem. In Proceedings of WTC 2019. 3rd World Transport Convention, 13.-16. Jun. 2019, Beijing, China.
2. *Mehler, C., Schiefelbusch, M.. Mobility on-demand: Disruption oder Hype? Entwicklung und Zukunft von Rufbus, Sharing und Robotaxi*,. In Der Nahverker 7+8/2017, DVV Media Group GmbH
3. . . . Still to do: Will be added.