Optimal Management of Railway Perturbations by Means of an Integrated Support System for Real-Time Traffic Control

Egidio Quaglietta [a, i, 1], Rob M.P. Goverde [a], Thomas Albrecht [b,h], Birgit Jaekel [b], Grégory Marlière [c], Paola Pellegrini [c], Joaquin Rodriguez [c], Twan Dollevoet [d], Bruno Ambrogio [e], Daniele Carcasole [e], Marco Giaroli [f], Gemma Nicholson [g]

[a] Department of Transport & Planning, Delft University of Technology, The Netherlands
[b] Dresden University of Technology, Germany
[c] IFSTTAR, Université de Lille Nord de France, France
[d] Erasmus University Rotterdam, The Netherlands
[e] NTT Data, Rome, Italy
[f] Ansaldo STS, Genoa, Italy
[g] Centre for Railway Research and Education, University of Birmingham, UK
[h] CSC Deutschland GmbH, Dresden, Germany
[i] Control Command & Signalling division, Network Rail Ltd, Milton Keynes, UK
[1] E-mail:e.quaglietta@tudelft.nl, Phone: +31 (0)15 27 82761

Abstract

Automatic real-time control of railway traffic perturbations has recently received the attention of practitioners. The aim is to make use of mathematical algorithms to maintain the required service availability during unplanned disturbances to operations. In the literature many tools for real-time traffic control are proposed, but their effects on traffic have never been studied neither in real life nor in realistic simulation environments. We can mention only a few pilot tests and a unique installation in the Lötschberg Base tunnel in Switzerland, which is in any case an ad-hoc implementation not extendible to other case studies. In this paper we present the ON-TIME framework for the real-time management of railway traffic perturbations. The main innovation is a standard web service-oriented architecture that ensures scalability and flexibility. A standard RailML interface is used for the input/output data of the modules, allowing immediate applicability of the framework to any network having a RailML representation. The scalability makes the framework independent from the number of modules and the amount of data exchanged. The flexibility permits any module to be replaced with others having similar features. The framework is tested in a closed-loop with the simulation environment HERMES for a perturbed traffic scenario on the Swedish Iron Ore line. Tests are performed for two different replanning algorithms (ROMA and RECIFE) used as conflict detection and resolution modules of the framework. The analysis represents a proof-of-concept to confirm the effectiveness of our framework in automatically solving conflicts and deadlocks during perturbed traffic conditions.

Keywords

Railway traffic control, real-time replanning, traffic optimization, standard communication architecture.

1 Introduction

The recent growth in the demand for railway transportation has resulted in a high traffic density and heavily used networks, which are sometimes working in saturated conditions. In this context, perturbations to traffic (e.g. extensions of running and/or dwell times) can lead to track conflicts, i.e. situations where two or more trains request the same block section in overlapped time periods. Track conflicts force trains to slow down or even stop at unplanned restricted signal aspects, thereby deviating train services from the original plan.

Slight perturbations can still be absorbed by time allowances in the timetable, but larger disturbances need to be specifically managed by replanning the service in real-time. Real-time replanning means adjusting the space-time trajectory of trains based on the current traffic information, with the aim of mitigating the impact of perturbations as much as possible. Basically a train can be replanned with control measures such as: changing the passage order at a given station or junction (reordering), modifying the arrival/departure times at a station (retiming), or even detouring the train over a different route (rerouting). The set of control measures that is planned to be taken in a given time period ahead is called the Real Time Traffic Plan (RTTP). A RTTP therefore contains the list of passage orders, arrival/departure times and/or routes that are planned to be respected by trains in the next time period. In other words the RTTP is the microscopic train path plan resulting when control measures are taken.

So far, the control measures contained in the RTTP are decided by human dispatchers on the basis of their own experience and/or rules-of-thumb. Nevertheless, it is very difficult for a human being to understand the effects of his/her decisions on traffic, especially in the case of large networks or heavily congested areas. This can sometimes result in control measures that may be not effective or even counterproductive. To this end advanced decision support systems have been proposed that automatically compute a set of control measures (i.e. a RTTP) that optimizes given traffic performance (e.g. minimizes the total delay, maximizes the punctuality) while ensuring conflict-efficient train operations. Conflict-efficient means that we aim to remove all track conflicts, but we cannot guarantee that trains will run without any conflicts once these measures are put into operation. Some trains could indeed still encounter restricted signal aspects during real operations. Train operations are optimized by a Conflict Detection and Resolution module (CDR), which consists of mathematical models for both detecting and solving track conflicts. Briefly, the Conflict Detection element considers current traffic information to predict future traffic conditions and detect potential track conflicts. The Conflict Resolution part solves the detected conflicts by identifying a set of control measures that optimizes a given objective function and allows conflict-efficient train operations.

Although several CDR models are proposed in literature ([3], [8], [20], [25]) nothing more than pilot tests can be mentioned ([14], [15], [17]). Automatic replanning tools have not been seriously applied into practice yet, mostly because infrastructure managers are afraid of implementing systems whose impacts on traffic are blurry and not well known. On the other hand, it is not clear yet how these systems can interface with real traffic, whether a standard communication interface can be defined and if these tools work for any traffic condition.

A concrete reply to these issues is provided by the European FP7 funded project ON-TIME [19]. A relevant part of this project focussed on designing, developing and testing an integrated framework for the optimal real-time management of railway traffic

perturbations. This paper describes the main outcomes of this research explaining the different modules of the framework and their interactions. A proof-of-concept is given that shows how traffic perturbations can be optimally and automatically managed by mathematical algorithms connected to operations through standard software interfaces. Many are the contributions to the literature provided by this paper, specifically:

- A perturbation management framework has been developed that integrates algorithms for traffic state monitoring, prediction, track and connection conflict detection and resolution, automatic route setting, driver advisory system.
- The algorithms of the perturbation management framework have been interfaced and tested within simulated operations in a closed-loop control.
- A web service-oriented architecture is realized which lets the algorithms communicate with each other and with simulated operations, in a standard, flexible software interface. Modularity allows replacement of any module with similar ones.
- A standardization of the data flow communicated among the different modules and with the simulated operations have been realized by using RailML [24].
- A novel formalisation has been developed for a XML representation of the current traffic state, the RTTP and the Train Path Envelope.
- A touch screen Human-Machine Interface has been connected via the architecture to both the framework and the simulated operations allowing dispatchers to visualize current and replanned traffic operations.

The framework executes closed loop control of railway traffic by following a rolling horizon approach. Current traffic information (e.g. train positions and speeds) is gathered from the field at regular time intervals (called Replanning Interval, RI) to automatically compute an optimal RTTP which tackles all conflicts detected in a certain time period ahead (called Prediction Horizon, PH). The resulting RTTP is then shown to the dispatcher for acceptance by means of a Human-Machine Interface. In cases where the dispatcher accepts the RTTP, this latter is implemented into the field and followed by train services.

The framework is tested versus real test cases within the simulated traffic environment HERMES. The ON-TIME project considered three railway networks across Europe: the East Coast Main line in the United Kingdom, the Iron Ore line in Sweden and the Utrecht-Eindhoven-Tilburg-Nijmegen corridors in the Netherlands. For each network several disturbed scenarios have been examined, including train entrance delays and infrastructure limitations such as temporary speed limit restrictions. Obtained results prove that our framework for real-time traffic management works effectively regardless of the network or the traffic conditions analysed. For the sake of brevity, this paper illustrates how the framework works when applied to a case in which a train is heavily delayed on the Iron Ore line.

The paper is structured as follows: a review of methods for replanning railway traffic in real-time is given in Section 2, while a description of our framework is provided in Section 3. Section 4 gives more details on each module composing the framework, while Section 5 illustrates how the framework works when applied to a real case study on the Swedish Iron Ore line. Conclusions and directions for future research are provided in Section 6.

2 Literature Review

In the literature several models for real-time replanning of railway traffic can be found.

We can mention macroscopic models such as those introduced by Carey and Lockwood [4], or Higgins et al. [11] to reduce train tardiness and/or energy consumption. Meng and Zhou [16] present a macroscopic stochastic programming approach to study the robustness of a meet-pass plan for a disrupted single track line. Chen et al. [5] introduce a macroscopic mixed-integer programming approach to reschedule trains at junctions and bottlenecks by means of a differential evolution algorithm. All these macroscopic models represent the network with a low level of detail, which is why they solve conflicts at the level of corridors between two consecutive stations or junctions. More accurate are instead microscopic approaches that detect and solve conflicts at the level of block sections taking into account constraints deriving from the signalling and Automatic Train Protection (ATP) systems as well as those regarding the detailed network topology (e.g. switches, platform layout). Microscopic models can be distinguished according to the formulation and the algorithm used to solve the replanning problem. Some authors such as D'Ariano and Pranzo [8] or Mazzarello and Ottaviani [15] base their models on alternative graphs. Törnquist [25], Pellegrini et al. [20] or Caimi et al. [3] adopt instead a Mixed-Integer Linear Programming approach.

The main limitation with all these approaches is that they have scarcely been tested in a closed-loop interface with real operations, nor with simulation environments that reproduce realistic traffic dynamics. Lüthi [13] defines in a schematic way how these replanning models could be integrated with real traffic to achieve closed-loop control. Such a scheme has been applied by Quaglietta et al. [21], [22] who interfaced the replanning model ROMA with the traffic simulation model EGTRAIN to study the stability and the quality of optimal RTTPs. The framework applied is however not scalable for implementation in real-life. To the best of the authors' knowledge the only real-life installation of a system for the automatic closed-loop control of railway traffic is the one in the Lötschberg Base tunnel in Switzerland [17]. This system builds on an ad-hoc framework that has the limitation of not being extendible to different networks or traffic conditions. Other real-life implementations do not go beyond pilot tests such as those described by Mannino and Mascis [14] or Mazzarello and Ottaviani [15]. It is clear that the gap between literature and practice stays on one hand in the definition of a standard framework for real-time traffic control that could be applied to any railway network and traffic conditions. On the other hand there is the necessity of proving by means of experiments in simulation that such a framework leads towards a perturbation management that is better than current practice. Filling this gap would motivate infrastructure managers to implement these systems into practice. This paper mainly contributes to filling this gap, providing a proof-of-concept, proving that such a system works and is implementable into real-life.

3 The ON-TIME Framework for the Optimal Real-Time Control of Railway Traffic

The ON-TIME framework developed for the optimal real-time management of railway traffic perturbations is illustrated schematically in Figure 1.

The modules of the framework communicate with each other by means of a web service-oriented architecture that works according to the principle of event publishing/subscription. This means that the output events from each module are published and queued in the architecture. An event stored in the architecture is then sent as input to all the modules that subscribed to that specific event. For instance if a module

returns as output an event type *A* this is published and queued in the architecture where it is then dispatched as input to all the modules that subscribed to the event type *A*. In the figure, the events published by each module are represented by the arrow directed towards the architecture, while those in input are depicted by the arrows in entrance to the module.

Railway traffic is represented by the microscopic simulation environment HERMES. This simulation model accurately reproduces all the dynamic interactions among the trains, the signalling/ATP systems (e.g. signals, braking behaviour), the infrastructure elements (platforms, switches) and the interlocking (e.g. dependencies between switch positions and signal aspects). Each time that a train occupies or releases a track detection section (e.g. track circuit) the corresponding event is published to the architecture. Each "track occupation/release" event is then forwarded to the Traffic State Monitoring (TSM), which elaborates these events to produce the current traffic state as output, i.e. the current position and speed of every train on the network. The current traffic state is transferred under the form of RailML to the architecture and then communicated to the Perturbation Management Module (PMM). The PMM is the core of the framework since it provides the control measures that allow the optimal real-time control of traffic perturbations. As can be seen the PMM is composed of three interacting sub-modules, namely the Traffic State Prediction (TSP), the Conflict Detection and Resolution (CDR) and the Connection Conflict Detection and Resolution (CCDR).

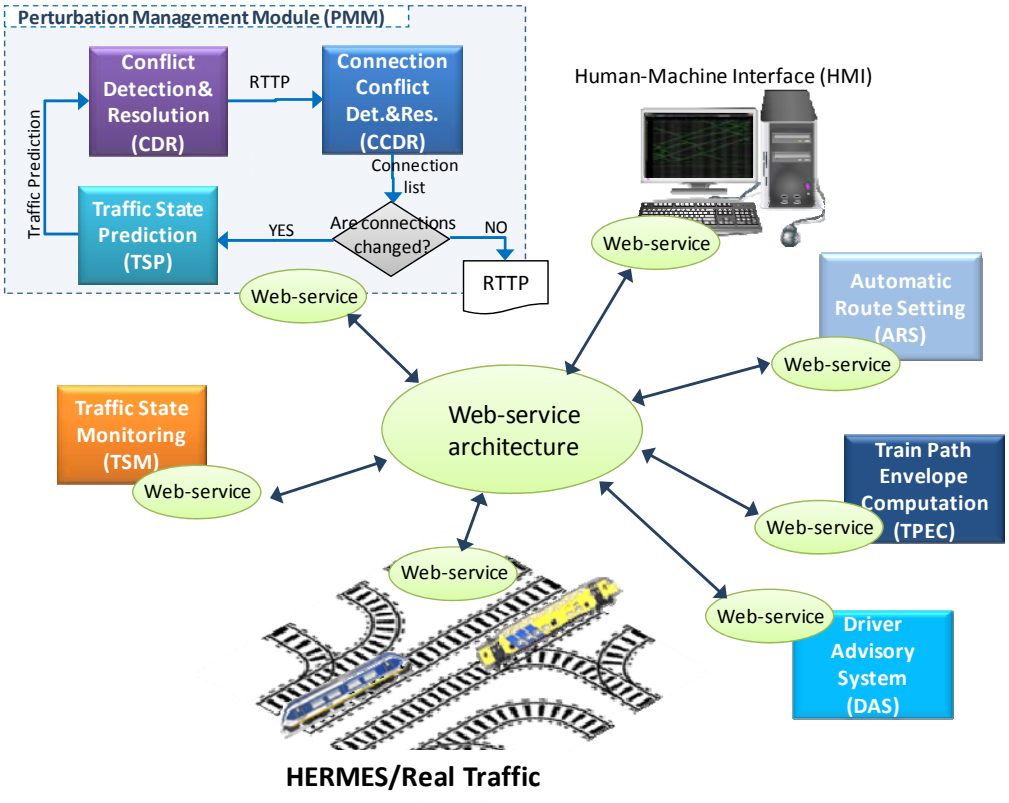

Figure 1: The ON-TIME framework for the real-time management of railway traffic perturbations.

The TSP receives as input from the architecture the current traffic state in order to forecast the traffic behaviour in a given time period ahead (the Prediction Horizon, PH). The traffic prediction is then set as input to the CDR module. Specifically, the Conflict Detection uses the prediction to identify track conflicts potentially occurring within the Prediction Horizon. If conflicts are detected the Conflict Resolution algorithm determines a set of control measures (i.e. reordering, retiming and/or rerouting), which optimizes certain traffic performance while guaranteeing conflict-efficient train operations. The control measures are printed out in the form of a Real-Time Traffic Plan expressed by means of a specifically designed XML scheme. This XML scheme has been appositely elaborated within the ON-TIME project to standardize the way in which dynamic railway data are expressed. The CCDR analyses the RTTP received as input, to identify all those connections that should be removed because they are critical in terms of delay propagation. The output of the CCDR is therefore a list containing all the connection constraints that should be removed to avoid delays. If no connection is cancelled, then the RTTP can be sent to the architecture. Otherwise, a new traffic prediction must be performed, taking into account the new list of connection constraints and a new RTTP must consequently be computed. The RTTP is communicated together with the traffic prediction to the Human-Machine Interface, which graphically shows to the dispatcher the optimal control measures computed by the CDR. If the dispatcher accepts these measures then the RTTP is implemented into the field.

Within the ON-TIME project we did not focus on the interaction with the human dispatcher, so we consider that every RTTP is automatically put into operation without the acceptance of the dispatcher. The implementation of the RTTP into the field is realized by the Automatic Route Setting (ARS). This module automatically implements train routes in the HERMES simulator, in the same order as established by the RTTP. In this way, trains running on the network follow the passage orders, the arrival/departure times and the routes contained in the real-time traffic plan. The route setting commands are sent to the simulator via the architecture, in the form of events. Based on the RTTP, the Train Path Envelope Computation (TPEC) calculates the buffer times between trains that can be exploited to fine-tune train speed-distance trajectories in an energy-efficient fashion. The train path envelopes are received as input by the Driver Advisory System (DAS), which determines energy-efficient train speed-distance trajectories that minimize the energy consumption while guaranteeing the respect of the scheduled arrival/departure times. The output of the DAS is then transferred through the architecture to HERMES where trains will follow the energy-efficient trajectories provided as input.

The real-time control of railway traffic follows a rolling horizon approach; this means that after every Replanning Interval the current traffic state is sent to the PMM and a new RTTP is computed and implemented into the field.

The adopted web-service architecture ensures independence amongst the modules and scalability with respect to the amount of data exchanged. This allows replacement of any module with another having similar characteristics. Such a framework can be straightforwardly applied in real-life if we replace the HERMES simulator with a real railway network. A relevant innovation is given by the standard communication interface achieved by expressing all the input/output data in standard RailML format. This feature enables immediate application of the framework to any railway network and traffic condition as long as the infrastructure, rolling stock, timetable and interlocking data are available in RailML.

4 Functional Description of the Modules

In the following we provide a concise description of all the modules composing the presented framework for real-time replanning. For each module we describe the input and output data, as well as the underlying mathematical models.

4.1 HERMES: A Microscopic Simulation Model of Railway Traffic

HERMES [10] is an object-oriented microscopic model for the simulation of railway traffic, developed in Java by the British company GRAFFICA. The model is based on six integrated components respectively representing the infrastructure, the rolling stock, the timetable, the signalling/ATP systems, the interlocking and the train driver behaviour. The infrastructure is described at a high level of detail and formalized as a directed graph. The nodes of the graph can represent signals, stop posts, the joints in between two track detection sections or the points of a switch. The arcs instead constitute track detection sections of the network. The rolling stock is modelled by considering all the characteristics of the vehicles such as the length, the mass, the number of coaches, the tractive effort-speed curve, the braking rate, and the resistance parameters. The timetable contains all the scheduled arrival/departure times at stations as well as the passage times at relevant signals and/or junctions. The signalling/ATP systems are modelled by considering the aspects of every signal and the corresponding ATP speed codes, which ensure a safe braking of the trains at red signals. The interlocking describes each route by means of the start and target signals, the sequence of track detection sections to be traversed and the direction required by the switches to set that route. The interdependencies between signal aspects and switch direction are also considered to avoid the simultaneous setting of conflicting routes. The driver behaviour component allows the specification of different parameters which enable a more aggressive rather than a more cautious driving style. This component has additional features that let the trains drive according to the scheduled speed-distance trajectories or follow energy-efficient driving advice provided by DAS algorithms. The simulation is performed by means of a hybrid approach where some components (e.g. trains) follow a time-driven simulation, while others (e.g. signals, switches, routes) are simulated on an event-driven basis. In particular, train movements are reproduced by integrating over time Newton's differential equations of motion. The integration is made numerically by using Simpson's quadrature rule [26]. The outputs consist of train speed-distance or time-distance trajectories as well as simulated train delays at stations. The HERMES simulator has an interface for the RailML export of all the network data. Moreover it is equipped with an open set of APIs that allow the user to introduce traffic disturbances, infrastructure disruptions and/or customise functions relative to the driver behaviour or the route setting.

4.2 The Traffic State Monitoring

The Traffic State Monitoring aims to collect all the track occupation/release data from the simulated traffic environment in order to estimate the current traffic state, i.e. the current position and speed of each train on the network. The estimation of the current traffic state is performed by adopting the approach of Albrecht et al. [1] who use the kinematic equations of motion to reconstruct train speed-distance trajectories based on track occupation data. The algorithm uses a least-squares optimization approach to estimate the train speed at each track section border and has been specially fitted for real-time application. It uses the last five section occupation events of a train (if available) but not less than three. If a planned or unplanned stop is detected (average speed on a track circuit

smaller than 10 km/h), the current speed is estimated as 0. If no speed estimation is possible, the speed is estimated to be 50% of the maximal permitted speed value. An extension of the algorithm has been made, which improves its behaviour in iterative calls like in our framework. Here, the information that a train has not yet entered the next track detection section at a given time could be an indicator that a train might have to brake. This is considered using an extension of the objective function of the optimizer. The whole module has been implemented in Java using the Apache math implementation of the Levenberg-Marquardt algorithm for least squares optimization.

4.3 The Traffic State Prediction

The aim of the Traffic State Prediction is to forecast time-distance and speed-distance trajectories of trains within the Prediction Horizon. Input data to this module are both static and dynamic. Static data describe the characteristics of the infrastructure (e.g. switches, platforms, track length and gradient), the timetable (e.g. scheduled arrival/departure times at stations), the rolling stock (e.g. mass, tractive effort-speed curve), the signalling/ATP systems (positions of signals, signal aspects and braking behaviour) and the interlocking. The dynamic data is the current traffic state produced by the TSM. The static input data are provided in standard RailML format, while the current traffic state is expressed with an XML appropriately defined. Time-distance and speed-distance trajectories of each train are predicted by numerically integrating Newton's equations of motion, considering current train speeds and positions as boundary conditions. The outputs are time-distance and speed-distance trajectories of each train running within the Prediction Horizon. The TSP module is written in Java and is generally separated from the Conflict Detection and Resolution module. In such a case the traffic prediction must be transferred as input to the CDR in order to detect and solve potential conflicts. Some CDR algorithms such as ROMA instead have the TSP directly integrated with the Conflict Detection model. In the case of ROMA, the TSP is coded in C++ and the predicted train operation times (running, headways and dwell times) are directly used for the conflict detection process.

4.4 The Conflict Detection and Resolution

This module is composed of two components: the Conflict Detection and the Conflict Resolution. The former detects the presence of potential track conflicts within the Prediction Horizon. The latter instead solves detected conflicts by identifying a set of control measures that optimize a given objective function while allowing conflict-efficient and deadlock-free train operations. We consider two different CDR models, namely ROMA [8] and RECIFE [20], which have been alternatively tested within our ON-TIME framework. The modularity of the framework allows the substitution of modules with similar characteristics; that is why we could separately test one or the other CDR model, just by plugging one or the other into the framework. Input data of both ROMA and RECIFE are the current traffic state and the microscopic characteristics of the infrastructure (e.g. positions of switches, platforms, track detection sections), the rolling stock, the signalling/ATP and the interlocking systems as well as the timetable.

The ROMA model uses alternative graphs to formulate both the conflict detection and the conflict resolution problems. More details about the mathematical formulation can be found in [8]. In particular, track conflicts are detected by means of blocking time theory [9], which considers a track conflict as an overlap between the blocking times of two different trains over the same block section. The conflict resolution instead is defined as an iterative two-step optimization problem. The first optimization problem considers train

routes as fixed and uses a truncated Branch and Bound algorithm [7] to identify the train passage orders and the shifts in departure/arrival times that minimize the maximum consecutive delay on the network. This latter is indeed the objective function adopted by the ROMA model. Once the optimal orders and times are found, the second optimization problem employs a tabu search [6] to identify possible alternative routes able to further reduce the maximum consecutive delay. If more convenient routes are found, another iteration of the two-step optimization is performed until no further improvement of the objective function is found or a computation time threshold is reached.

The RECIFE module tackles the conflict detection and resolution problems through the solution of a mixed-integer linear programming formulation [20]. This solution can be obtained through any integer programming solver. Namely, IBM ILOG CPLEX (version 12) showed extremely good performances on rather different case studies. The two problems of conflict detection and resolution are considered concurrently: if no conflict is detected, RECIFE immediately returns the original traffic plan. If conflicts are detected, they are solved to optimality, for what concerns both the rerouting and the rescheduling problems. If the optimality proof is not possible within a computation time threshold, the best traffic plan found within this time is returned. The quality of a traffic plan is assessed in terms of total delay of all trains at any station where they have a scheduled stop, plus the trains' delay at their exit from the network. This sum is the objective function optimized by RECIFE. The network is represented by taking into account the microscopic data up to the level of detail of track detection sections. Such a representation allows the modelling of both the route-lock-route-release and the route-lock-sectional-release interlocking systems. In our experiments, we considered the route-lock-sectional-release.

Both ROMA and RECIFE are developed in C++. The output of the CDR models is a Real-Time Traffic Plan containing the train passage orders, the shifts in the departure/arrival times and the routes that optimize the objective function.

4.5 The Connection Conflict Detection and Resolution

Train delays might severely affect the travel times of passengers in the system. This holds particularly for passengers who have to transfer from a so-called feeder train to a connecting train as part of their journey. When the feeder train is delayed and the time between the arrival of the feeder train and the departure of the connecting train is short, passengers might miss the connecting train. As a consequence, they have to wait for the next train to their destination, which significantly increases their travel time. In such cases, it can be beneficial to delay the departure of the connecting train, too, to allow these passengers to catch the connecting train. The aim of the Connection Conflict Detection and Resolution module is: to detect such connection conflicts and decide for each connection conflict whether the connecting train should wait for the delayed passengers or should depart on time.

The CCDR module takes the RTTP as input from the CDR and information about the travel plans of all passengers in the system. It then detects any connection conflicts that might be present and solves a mathematical optimization model to determine which connections can be dropped. The objective of this module is to minimize the total delay of all passengers in the system. The CCDR module is written in Java. The output of the module is a list of connections that can be dropped.

4.6 The Human-Machine Interface

The Human-Machine Interface (HMI) is a module written in Java composed of two sub-modules, the Line Describer (LD) and the Train Graph (TG). The LD uses the

infrastructure data expressed in RailML to visualize on the screen the network topology including all the infrastructure elements like platforms, switches and signals. The user is also allowed to stretch or rotate graphical elements on the screen for adjusting the network visualization to his/her specific needs.

The TG instead illustrates both the planned and actual time-distance diagram of all train services. The planned time-distance diagrams can be given by either the RTTP or the original timetable (when no RTTP has been computed yet by the framework). Both the RTTP and the timetable must be expressed in RailML format.

The actual time-distance diagrams realized by trains until the current time are instead obtained directly from track occupation/release events provided by the simulated environment HERMES. Planned and actual time-distance diagrams are respectively represented with a thin and a thicker line.

4.7 The Automatic Route Setting

This Java module implements the routes in the simulation environment by issuing route setting commands in the same order as established by the RTTP. Route setting commands are issued at the "most suitable" time. On one hand, this time is early enough to prevent trains meeting restricted signal aspects due to too late a route setting. On the other hand it is late enough to be capable of changing route orders for as long as possible, so as to allow more flexibility in the traffic management. Figure 2 shows the time-distance diagram predicted by the TSP (blue solid line), the start signal *S0* of route *R0*, the automatic block signals *S2* and *S1*, as well as the predicted arrival times at these signals, t_2 and t_1. The most suitable time t_0 to set route *R0* is obtained by subtracting from time t_1 the signal sight time t_S, the driver's reaction time and the time to set the route T_{RD}.

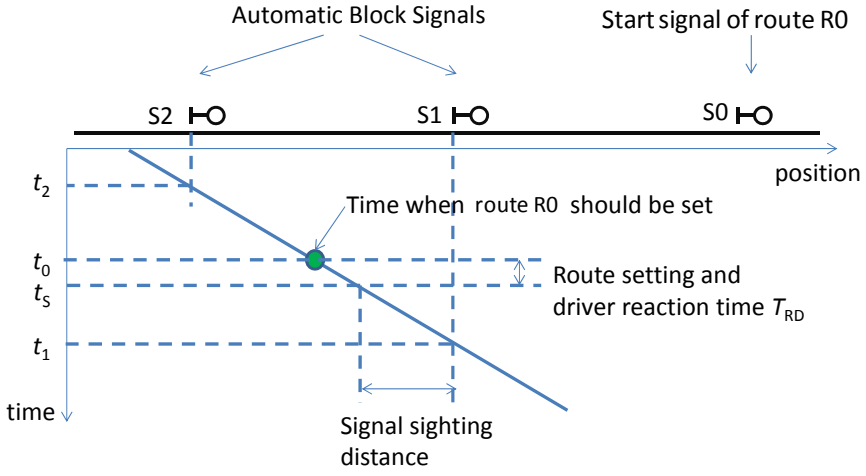

Figure 2: Calculation principle of the most suitable route setting time.

A route can be set if this is available on the infrastructure side (Condition A) and if the most suitable time t_0 is passed (Condition B). Both conditions are checked by algorithms running in parallel and lead to any of the following three states of a route: *i*) not all route elements are available (State 0); *ii*) all route elements are available (State 1); *iii*) the route is actually set for a train (State 2).

Condition A is fulfilled for a route when no train occupies a track detection section belonging to this route and no train has reserved the route. As soon as condition A is fulfilled, the state of the route switches from 0 to 1 and condition B is checked. If condition B is also satisfied, the route passes immediately to state 2, otherwise it remains in state 1.

When the RTTP is updated by the CDR module all routes that are in state 2 will be reset to state 1 and condition B will be checked again.

4.8 The Train Path Envelope Computation

A Train Path Envelope (TPE) is a sequence of time windows in which trains can drive in an energy-efficient way without hindering the next train and/or being hindered by the previous one. In order to avoid conflicting train operations, the DAS must define energy-efficient train paths that are contained in these windows. For a given train, the TPE is computed by taking into account the buffer times with the previous and the next train. This requires the computation of blocking times for all trains, considering the sequence of train services and the running times scheduled by the RTTP. Inputs to this module are the RTTP, from which train sequences for all track detection sections are extracted, and a database containing information about the energy consumption of different train types on the track detection sections. We first combine all subsequent track detection sections being traversed by the same sequence of trains (CTDS) in order to lower the number of variables in the computation. Then the TPE computation is performed in three steps: *i*) train movements are microscopically simulated for each CTDS of each train; *ii*) energy optimal train paths are computed for each train considering as constraints the train sequence on the CTDS and the driving times given by the previous step; *iii*) blocking times on every track detection section are computed for each train assuming the train paths provided at step *ii*).

The output of this module is an XML dataset that contains for each train the time windows that can be exploited for every track detection section to minimize energy consumption while avoiding track conflicts. The whole module is developed in Java. A more detailed description of this module is given in [12].

4.9 The Driver Advisory System

The concept of centrally guided train control has been defined in the ON-TIME project, which has as its main component a driver advisory system that enables the driver to follow the train path envelope along an energy-optimal trajectory computed therein.

The main algorithmic component of such a concept is the computation of energy-optimal train trajectories that are drivable by a human driver, i.e. that contain a limited number of changes. Furthermore, the algorithm should cover a wide range of applications; therefore traction specifics like engine efficiency or energy recovery from braking are not considered. Here, an existing algorithm based on the regimes obtained through application of the maximum principle and the application of an iterative gradient method [2] has been extended to consider the additional restrictions of the train path envelope. The extensions mainly consider the introduction of a new kind of optimization entity (target window section limited by two consecutive target windows) and the process to find the optimal times and speeds at the position of these target windows (within the restrictions given by the perturbation management). This algorithm delivers the trajectory of the train as a sequence of points along the train run that contain information about time, speed, distance and driving regime. The implementation of this module is realized in Java. Different architecture alternatives have been developed, where the computation of the trajectory can

either be directly on-board or in a central unit and then be transmitted using XML data formats to the on-board unit.

4.10 Standardized Static and Dynamic Input/Output Data

Static and dynamic input/output data exchanged among the different modules have been expressed in a standard format. Standardization makes data independent from the modules, allowing easy substitution of a module with others with similar characteristics. Having standard input/output interfaces also allows immediate applicability of the framework to any network whose data are expressed with the same standard. Specifically, static data regarding the infrastructure, the timetable, the rolling stock and the interlocking are input to the framework and are represented in the standard RailML version 2.2. The XML schemes of each one of these data types can be seen on the RailML official website [24]. Dynamic data like the current traffic state, the RTTP, the TPEC and the energy-efficient driving regimes of the DAS are represented with specifically designed XML developed during the ON-TIME project. These XML schemes are respectively represented in Table 1, Table 2, Table 3 and Table 4.

Table 1: Snippet of the standard representation of the current traffic state.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<trafficState currentTime="2013-10-21 01:02:00.000 CEST" xmlns="be.jdb.jaxb.model" >
<trainStateInArea>
    <trainID trainNumber="T00009919B"/>
    <trainPosition trackID="SEC_3" travelDirection="-1" posOnTrack="170.0"
        currentTrackVacancyDetectionSection="TDS00013"
        previousTrackVacancyDetectionSection="TDS00011" astOccupationTime="01:01:30 CEST"/>
    <speed>0.0</speed>
</trainStateInArea>
<trainStateInArea>
    <trainID trainNumber="T00009922B"/>
    <trainPosition trackID="SEC_58" travelDirection="1" posOnTrack="50.0"
        currentTrackVacancyDetectionSection="TDS00156"
        previousTrackVacancyDetectionSection="TDS00152" lastOccupationTime="01:01:20 CEST"/>
    <speed>27.0</speed>
</trainStateInArea>
```

Table 2: Snippet of the standard representation of the RTTP.

```
<rTTP>
   <rTTPTrainView>
    <rTTPForSingleTrain trainId="T00004005">
         <tDSectionOccupation tDSectionID="TDS00120" trainID="T00004005" occupationStart="01:02:00
             CEST" routeId="routeR00095"/>
         <tDSectionOccupation tDSectionID="TDS00112" trainID="T00004005" occupationStart="01:08:37
             CEST" routeId="routeR00081"/>
         <tDSectionOccupation tDSectionID="TDS00111" trainID="T00004005" occupationStart="01:08:59
             CEST" routeId="routeR00081"/>
     </rTTPForSingleTrain>
    </rTTPTrainView>
   <rTTPInfrastructureView>
    <rTTPForSingleTDSection tDSectionId="TDS00081">
         <tDSectionOccupation tDSectionID="TDS00081" trainID="T00009926B" occupationStart="01:02:07
             CEST" routeId="routeR00050"/>
         <tDSectionOccupation tDSectionID="TDS00081" trainID="T00009921B" occupationStart="01:39:49
             CEST" routeId="routeR00052"/>
         <tDSectionOccupation tDSectionID="TDS00081" trainID="T00004005" occupationStart="01:57:58
             CEST" routeId="routeR00052"/>
       </rTTPForSingleTDSection>
   </rTTPInfrastructureView>
 </rTTP>
```

The current traffic state reports the current time (*CurrentTime*) and represents each train on the network with the tag <*trainStateInArea*> specifying the train number (*trainNumber*), the section (*trackID*) and the track detection section currently occupied (*CurrentTrackVacancyDetectionSection*), as well as the track detection section occupied previously (*PreviousTrackVacancyDetectionSection*).

The RTTP contains the solution of the CDR as observed from two different points of view, namely the train and the infrastructure. The RTTP is indeed composed of two parts called “train view” and “infrastructure view”, respectively. The “train view” is identified by the tag *<rTTPTrainView>*. For each train (*trainID*) it reports the chronological sequence of track detection sections to be crossed, specifying the ID (*tDSectionID*), the time when the train is expected to enter them (*occupationStart*) and the route (*routeID*) the sections belong to. The “infrastructure view” instead is identified by the tag *<rTTPInfrastructureView>*. For each track detection section (*tDSectionID*), it gives the chronological sequence of the trains (*trainID*) crossing it, the corresponding entrance time (*occupationStart*) as well as the route (*routeID*) the section belongs to.

It is clear that these two parts of the RTTP are nothing but two different standpoints of the same CDR solution.

The “infrastructure view” is a more suitable representation to use for the ARS module, while the “train view” is preferred by the HMI for the visualization of train time-distance trajectories on the screen.

Table 3: Snippet of the standard representation of the TPE.

```
<tpe xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <corridor train_service_id="S3521" timestamp="10/12/2018 07:28:20">
    <corridorWindow position_id="startd15e30_Ut_Gdm40" time_min="90,00" time_max="90,00"
      vel_min="0" vel_max="50" nonCommercialStop="false" reason="reason:StartOfEnvelope" />
    <corridorWindow position_id="TDS0_Ht0026d15e1562_41080" time_min="1243,11" time_max="6901,83"
      vel_min="20" vel_max="80" nonCommercialStop="false" reason="reason: trainBeforeLeaves" />
    <corridorWindow position_id="TDS0_Ht0038d15e911_2205" time_min="1263,41" time_max="6957,06"
      vel_min="40" vel_max="120" nonCommercialStop="false" reason="reason: hostTrainGoSlower" />
    <corridorWindow position_id="TDS0_NHt111d15e1433_41090" time_min="1291,26" time_max="7023,06"
      vel_min="40" vel_max="120" nonCommercialStop="false" reason="reason: endOfEnvelope" />
  </corridor>
  <corridor ...../>
</tpe>
```

Table 4: Snippet of the standard representation of the DAS energy-efficient advice.

```
<trajectory id="traj1_S3523" obuId="127.0.0.1" timeStamp="07:31:54.832" trainId="S3523">
  <samplingPoint drivingRegime="ACCELERATION" optimalSpeed="0" position="193"
      reasonCode="energyOptimisation" reasonText="Ht" time="07:52:01.000"/>
  <samplingPoint drivingRegime="CONSTANTSPEED" optimalSpeed="40" position="298"
      reasonCode="energyOptimisation" reasonText="Ht" time="07:52:19.836"/>
  <samplingPoint drivingRegime="ACCELERATION" optimalSpeed="40" position="921"
      reasonCode="energyOptimisation" reasonText="Ht" time="07:53:15.892"/>
  <samplingPoint drivingRegime="CONSTANTSPEED" optimalSpeed="80" position="1657"
      reasonCode="energyOptimisation" reasonText="Ht" time="07:53:56.799"/>
  <samplingPoint drivingRegime="COASTING" optimalSpeed="80" position="2254"
      reasonCode="energyOptimisation" reasonText="Ht" time="07:54:23.625"/>
</trajectory>
```

The Train Path Envelope reported in Table 3 gives for each train service (*train_service_id*) the current time (*timestamp*) and the sequence of time windows computed for each corridor (*corridorWindow*). Corridors correspond to track detection sections, identified by means of the id of their borders (*position_id*). The time window relative to a track detection section is determined by the lower (*time_min*) and the upper (*time_max*) bounds, both expressed in seconds. Also the lower (*vel_min*) and the upper (*vel_max*) bounds of the energy-efficient speed advice are given to the DAS algorithm, together with the information on the presence of non commercial train stops (*nonCommercialStop*).

The XML scheme in Table 4 provides the energy-efficient speed advice as output from the DAS module. For each train (*trainId*) it is reported the current time (*timestamp*), the ID of the on-board unit (*obuId*) and the list of switching points (*samplingPoint*) among successive energy-efficient driving phases. For each switching point is it specified the

driving regime (*drivingRegime*), and the optimal speed in km/h (*optimalSpeed*), that must be reached at the curvilinear abscissa *position* (in metres) at the given *time*.

4.11 The Web Service-Oriented Architecture

The web service-oriented architecture enables the communication among the different modules of the framework. This is an event-based architecture where the data to be transferred from one module to another are considered as an event. The architecture identifies every event by means of a unique "type" identifier. Each module of the framework can link to the architecture as a "subscriber" or "publisher" for one or more type of event. A module is a publisher for a given event type when it produces that event as output. Instead it is a subscriber when it needs to receive that type of event as input. If a module is subscriber to a given event type it cannot receive events of different types unless it is also subscribed to them. We give a simple example to clarify the concept. The current traffic state produced by the TSM is seen by the architecture as an event of a given type, say type *A*. In this case the TSM is a publisher for the events of type *A*. The PMM instead is a subscriber for type *A* events since it takes the current traffic state as input. If an event of a given type cannot be dispatched to one or more of its subscribers, it is stored in a queue and transferred as soon as possible. The storage is realized by using the open-source non-relational database MongoDB [18] while events are routed by means of the message broker Rabbit MQ [23]. The modules interact with the architecture using a REST service interface, exposed by a .NET module. Stress-tests showed that more than hundreds of millions of messages per day can be efficiently handled by our architecture without any message being lost. The architecture is scalable since all its components are capable of scaling by adding new modules in the network. Since message payloads are not processed, this gives flexibility in adding new event types or modifying existing ones on the fly.

Data exchanged between modules is represented using the RailML standard. Such a feature allows faster integration of new modules and immediate applicability of the framework to any network that uses the RailML format.

5 Application to the Iron Ore Line in Sweden

The ON-TIME framework for the real-time management of traffic perturbations has been applied to real case studies.

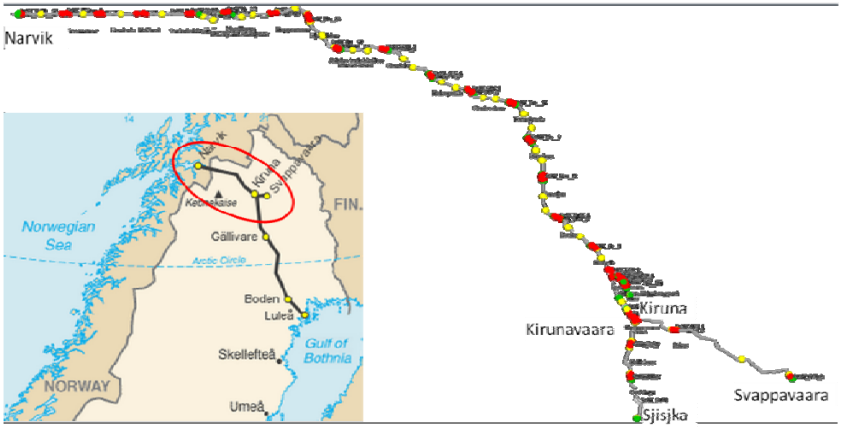

Figure 3: Narvik-Svappavaara corridor on the Iron Ore line.

As follows, we provide an illustrative example of the framework when applied to solve the case of a heavily delayed train on the Swedish Iron Ore line.

We refer in particular to the Narvik-Svappavaara corridor whose microscopic configuration is illustrated in Figure 3, together with its geographical location. This is a single track line crossing the border between Sweden and Norway. Twenty meet-pass points are distributed over the network, where trains moving in opposite directions can overtake each other. A total of 31 interlocking areas are present on the network, including the stations of Narvik (at the northern border), Kiruna, Kirunavaara, Svappavara (at the south-eastern border) and Sjiskja (at the southern border).

A mixed traffic pattern runs on this corridor, with a prevalence of freight trains over passenger ones. There are freight trains running between Kiruna and Narvik (and vice versa), which are 750 metres long and run at a max speed of 60 km/h, when loaded. Other freight trains run instead between Svappavaara and Narvik via Kiruna, and have a length of 600 metres with a maximum cruising speed of 100 km/h, when loaded. The local passenger trains operate on the corridor Narvik-Kiruna and vice versa, and run at a maximum cruising speed of 160 km/h. Some of the meet-pass points are shorter than the freight trains' length. For this reason, freight trains cannot meet trains running in the opposite direction at these points, otherwise deadlocks can occur.

We analyse a perturbed scenario which assumes that freight train 9904 from Kiruna to Narvik has an entrance delay of 40 minutes in Kiruna. This train enters the network at 03:09 a.m. instead of 02:29 a.m., as originally scheduled. We observe traffic for a total period of 7 hours from midnight to 07:00 in the morning. In this period a total of 15 trains are scheduled on the line, including 7 freight and 2 passenger trains directed towards Kiruna, as well as 5 freight and 1 passenger trains directed towards Narvik.

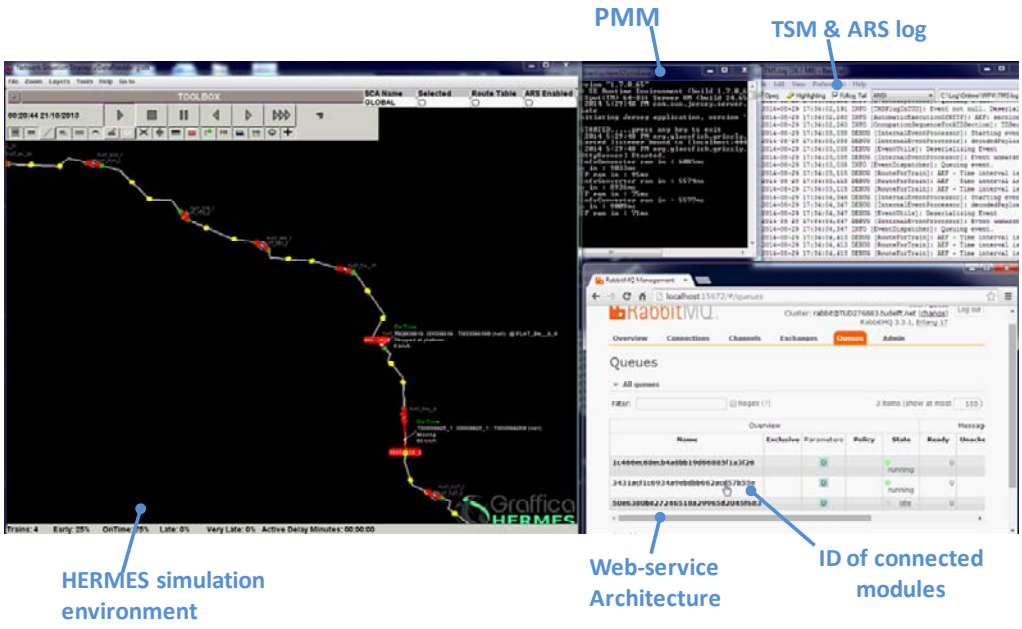

Figure 4: ON-TIME framework operating on the Iron Ore line.

By simulating the mentioned perturbed scenario in HERMES we observe that deadlocks occur when no specific dispatching action is taken and trains operate according to a simple First Come First Served rule. For this reason we apply our ON-TIME framework to optimally manage the perturbation, while allowing conflict-efficient and deadlock-free train operations. Given the prevalence of freight trains, no train connection is scheduled in the observed period. For this reason the CCDR module is not activated in this case. Our application mainly focuses on solving deadlocks and conflicts by means of optimal control measures; that is why we disregard energy efficiency by deactivating the TPEC and the DAS modules.

Figure 4 shows our ON-TIME framework actively working to solve the defined perturbed scenario on the Iron Ore line. Traffic is controlled in a closed-loop according to a rolling horizon approach with a replanning interval and a prediction horizon that we set to 2 minutes and 1 hour, respectively. This means that every 2 minutes the TSM (shown at the top-right) estimates the current traffic state, and a new RTTP is computed for the next hour, by the PMM (in the centre). The RTTP is then automatically implemented by the ARS (at the top-right corner) and followed by the traffic simulated in HERMES (on the left). The web-service architecture (at the bottom-right) enables the communication among all the modules, which are identified by means of an alphanumeric ID, as shown in the picture.

Each time that a new RTTP is produced, its “infrastructure view” is used by the ARS to implement the routes in HERMES, in the same order as given by the CDR solution. The “train view” is instead employed by the HMI to graphically visualize the RTTP on the screen.

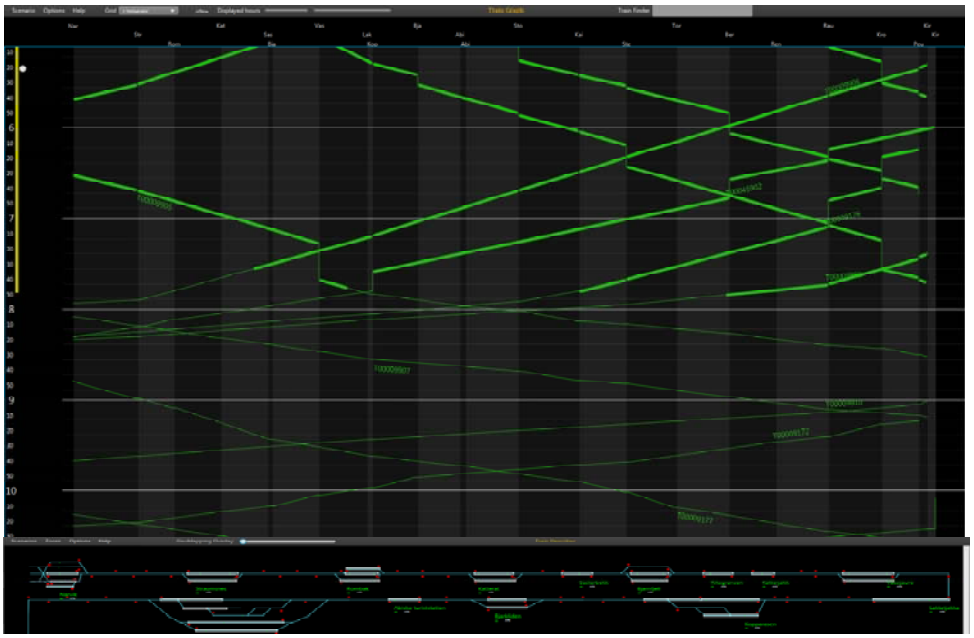

Figure 5: HMI showing the train path diagram and the network layout (at the bottom)

The HMI is illustrated in Figure 5 where the time is reported on the vertical axis and the distance is reported on the horizontal axis together with a schematic representation of the network (at the bottom). The thin green lines represent the time-distance trajectories of each train as scheduled by the RTTP. These trajectories will therefore be updated each time the CDR computes a new RTTP. The thicker green lines instead depict the time-distance trajectories actually realized by the trains until the current time. This information is derived directly from track occupation/release events sent by the simulation environment through the architecture. The HMI highlights that when the ON-TIME framework is applied to the perturbed traffic, trains run smoothly without any deadlock or conflict.

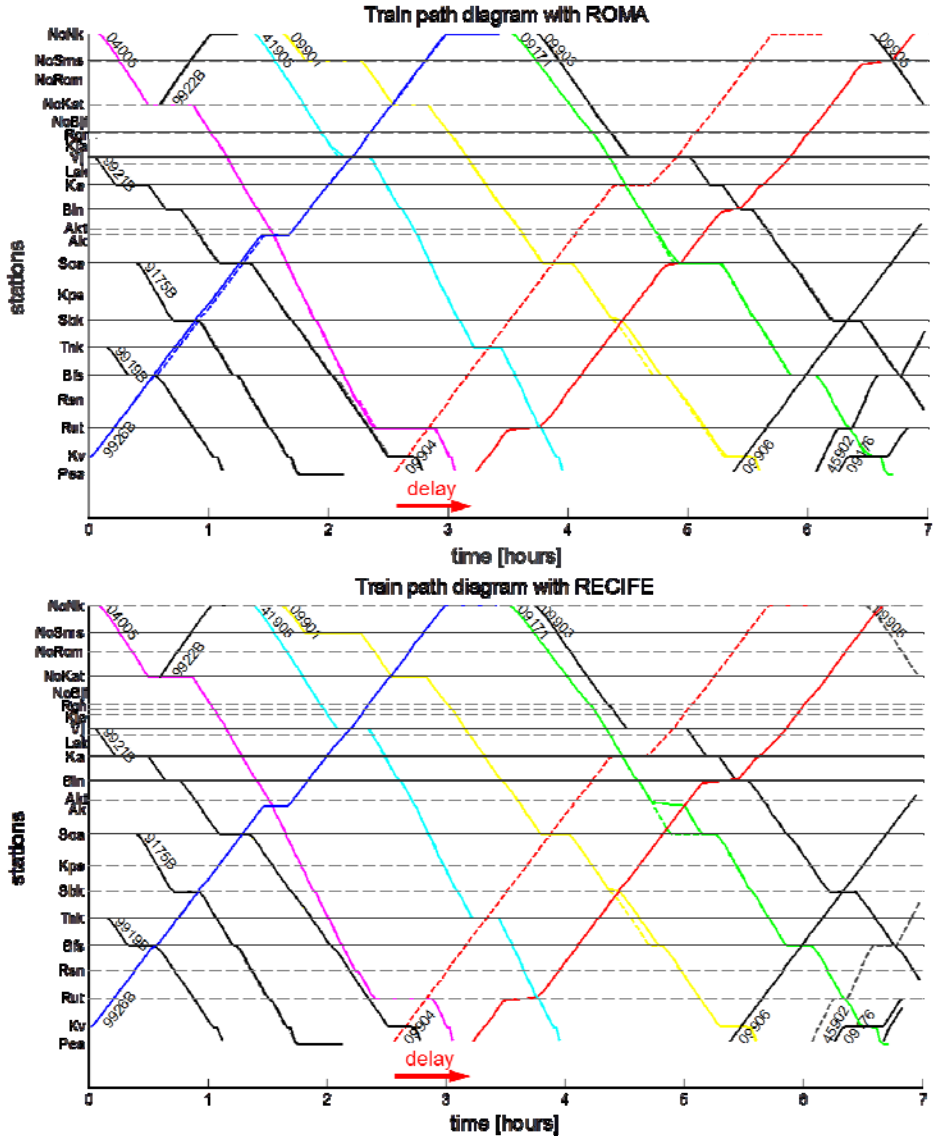

Figure 6: Train path diagram obtained by using ROMA (top) or RECIFE (bottom) as CDR.

We analyse the behaviour of the ON-TIME framework for two different configurations of the CDR module. The first configuration uses the ROMA algorithm, while the second employs the RECIFE algorithm. The perturbed scenario has been managed therefore by using each one of the configurations. Figure 6 shows the train path diagram obtained when the perturbed traffic scenario is managed by using ROMA or RECIFE as CDR of the framework. The vertical axis represents the distance while the horizontal axis reports the time. The dashed lines depict the time-distance trajectories as originally scheduled by the timetable. The solid lines are the time-distance trajectories realized when trains are controlled by the framework. The delayed train 9904 is reported in red and its entrance delay is underlined by the red arrow. As can be seen the ON-TIME framework is able to effectively tackle conflicts and avoid deadlocks independently from the CDR algorithm used. In this particular case the two algorithms manage the perturbation in a similar way. The main substantial difference stays in the meeting point between the delayed train 9904 and 9171, which runs in the opposite direction. With ROMA this meeting occurs at Stordalen (Soa) where train 9171 has a short planned stop. RECIFE instead establishes the meeting at Abisko Astra (Ak) where 9171 does not have any planned stop.

While ROMA prefers to keep on time train 9905 by forcing train 9904 to wait for it at Straumnsnes (NoSms), RECIFE avoids this meeting and delays the departure of 9905. Analogously, this happens for trains 45902 and 9176 which are delayed by RECIFE in the meeting with train 9171 running in the opposite direction.

Remark that, as mentioned in Section 4.4, the two algorithms optimize with respect to different criteria. Furthermore, in the closed-loop, both ROMA and RECIFE keep optimizing traffic until the end of the simulated time. Hence, in the last optimization performed, all the traffic between 6:58 and 7:58 is tackled by the algorithms, which then might tackle conflicts which are not observable in the train path diagrams shown.

6 Conclusions and Further Research

In this paper we present an innovative framework for the closed-loop control of railway traffic during perturbations. Several interacting modules compose the framework. The Traffic State Monitoring collects traffic information from the field to estimate the current traffic state. This latter is used by the Perturbation Management Module to compute optimal Real-Time Traffic Plans, based on traffic predictions made over a given prediction horizon. These plans are shown for acceptance to the human dispatcher by means of a Human Machine Interface. In case of acceptance the plans are then automatically implemented into the field by the Automatic Route Setting module, which sets train routes in the same order as established by the computed plan. Speed advice for energy-efficient driving is also produced and communicated to the trains by the Driver Advisory System module.

The main contribution our framework gives to both practice and to the literature, is the way the different modules communicate among each other. A standard, scalable and flexible web service architecture is responsible for storing and transferring data within the framework. The input/output data of every module have been standardized by expressing them in RailML format. Static data (e.g. infrastructure, rolling stock) uses the standard RailML language, while specific XML schemes are built up for expressing dynamic data (e.g. current traffic information and RTTP). This feature permits the immediate application of the framework to any railway network already represented in RailML.

Scalability ensures independence from the number of modules and the amount of data exchanged, while flexibility allows replacement of any module with another having similar characteristics. Thanks to this latter characteristic we were able to test the framework with different Conflict Detection and Resolution algorithms. Specifically the ROMA and RECIFE models have been adopted.

During the ON-TIME project the framework has been tested in a closed-loop with the traffic simulation environment HERMES for several European networks and perturbed scenarios. The tests show the general applicability of our framework, independently from the network topology, the traffic pattern and the perturbation considered. As an illustrative example this paper reports the test made on the Swedish Iron Ore line for a perturbed traffic scenario. The analysis performed constitutes a proof-of-concept to confirm that our framework is able to automatically solve conflicts and deadlocks during perturbations.

Future research aims at including in the control loop also the interaction with the human dispatcher by means of the HMI. Further developments will enable the implementation of the framework in real life.

Acknowledgement

The research leading to this paper was funded by the European Union's Seventh Framework Programme (FP7/2007-2013) in the ON-TIME project under Grant Agreement SCP1-GA-2011-285243.

7 References

[1] Albrecht T., Goverde R.M.P., Weeda V.A., Van Luipen J., Reconstruction of train trajectories from track occupation data to determine the effects of a Driver Information System, *Computer in Railways X*, pp. 207-216, WIT Press, 2006.

[2] Albrecht, T., Binder, A., Gassel, C.,: Applications of real-time speed control in rail-bound public transportation systems, *IET Intelligent Transport Systems,* Vol. 7 (3), pp. 305-314, 2013.

[3] Caimi G., Fuchsberger M., Laumanns M., Lüthi M., A Model Predictive Control Approach For Discrete-Time Rescheduling In Complex Central Railway Station Areas, *Computers & Operations Research*, Vol. 39, pp. 2578-2593, 2012.

[4] Carey, M., Lockwood, D., A model, algorithms and strategy for train pathing, *Journal of the Operational Research Society*, Vol. 46 (8),pp. 988–1005, 1995.

[5] Chen, L., Schmid, F., Dasigi, M., Ning, B., Roberts, C., Tang, T., Real-time train rescheduling in junction areas, *Proceedings of the IMechE: Part F – Journal of Rail and Rapid Transit*, 224(6), pp. 547-557, 2010.

[6] Corman F., D'Ariano A., Pacciarelli D., Pranzo M., A tabu search algorithm for rerouting trains during rail operations, *Transportation Research Part B*, Vol. 44 (1), pp. 175-192, 2010.

[7] D'Ariano A., Pacciarelli D., Pranzo M., A Branch And Bound Algorithm For Scheduling Trains In A Railway Network, *European Journal of Operational Research*, Vol. 183(2), pp. 643–657, 2007.

[8] D'Ariano, A., Pranzo, M., An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe

disturbances. *Networks and Spatial Economics*, 9(1), pp. 63-84, 2008.
[9] Hansen I.A., Pachl J., Railway Timetable and Traffic, Eurailpress, 2008.
[10] HERMES Simulator, http://graffica.co.uk/rail-traffic-management/simulation/, last accessed November, the 1st, 2014.
[11] Higgins, A., Kozan, E., Ferreira, L., Optimal scheduling of trains on a single line track, *Transportation Research Part B*, Vol. 30 (2), pp. 147–161, 1996.
[12] Jaekel B., Albrecht T., Interfacing Conflict Resolution and Driver advisory Systems in Railway Operations, *Proceedings of the 3rd international Conference on Models and Technologies for ITS*, pp. 333-343,TUD Press, Dresden, 2013.
[13] Lüthi M., *Improving the efficiency of heavily used railway networks through integrated real-time rescheduling*, PhD thesis, ETH Zurich, 2009.
[14] Mannino C., Mascis A., Real-Time Traffic Control in Metro Stations, *Operations Research* 57(4), pp. 1026-1039, 2009.
[15] Mazzarello, M., Ottaviani, E., A traffic management system for real-time traffic optimisation in railways. *Transportation Research B*, 41(2), pp. 246–274, 2007.
[16] Meng L., Zhou X., Robust Single-Track Train Dispatching Model Under A Dynamic And Stochastic Environment: A Scenario-Based Rolling Horizon Solution Approach, *Transportation Research Part B*, Vol. 45, pp. 1080-1102, 2011.
[17] Metha, F., Roessiger, C., Montigel, M., Latent energy savings due to the innovative use of advisory speeds to avoid occupation conflicts. *Computers in Railway XII*, WIT Press, pp. 99-108, 2010.
[18] MongoDB, http://www.mongodb.org/, last accessed, November, 12th, 2014.
[19] ON-TIME, http://www.ontime-project.eu/, last accessed July, 24th , 2014.
[20] Pellegrini, P., Marlière, G., Rodriguez, J., Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological*, 59, pp. 58-80, 2014.
[21] Quaglietta E., Corman F., Goverde R.M.P., Analysis of a closed-loop control framework in a realistic railway traffic environment, *Proceedings of the 3rd international Conference on Models and Technologies for ITS*, pp. 407-418,TUD Press, Dresden, 2013.
[22] Quaglietta E., Corman F., Goverde R.M.P., Stability of railway dispatching solutions under a stochastic and dynamic environment, *Journal of Rail Transport Planning & Management*, vol. 3(4),pp. 137-149, 2013.
[23] RabbitMQ, www.rabbitmq.com, last accessed 5th November 2014.
[24] RailML website, www.railml.org, last accessed 30th October, 2014.
[25] Törnquist J., Railway traffic disturbance management: An experimental analysis of disturbance complexity, management objectives and limitations in planning horizon. *Transportation Research Part A* 41(3), pp. 249-266, 2014.
[26] Ujevic N., New error bounds for the Simpson's quadrature rule and applications, *Computers & Mathematics with Applications*, Vol. 53(1), pp.64-72, 2007.