Enhancing City Transportation Services Using Cloud Support

Andrea Fornaia, Christian Napoli(✉), Giuseppe Pappalardo, and Emiliano Tramontana

Department of Mathematics and Informatics, University of Catania, Viale Andrea Doria 6, 95125 Catania, Italy
{fornaia, napoli, pappalardo, tramontana}@dmi.unict.it

Abstract. Smart cities will provide enhanced monitoring of crucial infrastructure resources, connectivity to users and advanced information services. Thanks to gathered data the quality of traditional services and infrastructures will be improved, and further services can emerge, such as novel urban transportation services. This paper devises a solution that enforces the cooperation between mobile devices and cloud infrastructures with the aim to bring public transportation where the people need it. Thanks to smart phones, sensing user locations, a request for transportation vehicles can be sent to a cloud-based intelligence, which filters and serves requests according to available transport routes, and their adaptation to user needs. Then, the available transportation vehicles will be timely alerted to operate accordingly.

Keywords: Cloud computing · Mobile users · Workflow · Monitoring · Intelligent assistive services · Smart cities

1 Introduction

We are observing a growing interest in the area of Smart Cities and services providing citizens with means to better organise daily life, and Smart Mobility is one of the most promising topics concerning future city environments (Benevolo et al. 2016). The ultimate goal is providing enhanced urban services, by means of information technology, to obtain a positive impact on the quality of life in urban areas. Major efforts will be devoted to the monitoring of critical infrastructures, the interconnection of physical and social infrastructures, as well as supports to assist physical infrastructures and, of course, citizens (Batty et al. 2012; Chourabi et al. 2012). This paper analyses the scenario of public transportation in urban areas and provides a solution to organise the flow of publicly accessed vehicles according to user requests, i.e. on-demand. Moreover, connectivity to users is enhanced according to the monitored transportation needs. Surely, the communication infrastructure is crucial to ensure the right timing in the planning of routes for the vehicles. The devised solution paves the way for future autonomous unmanned vehicles.

In such a scenario, a user request for a service aiming to find or bring a vehicle closer to her/him has to be taken into account by a defined *workflow*, i.e. a standard flow of execution for the activities related to the offered service (Fornaia et al. 2015).

G. Dregvaite and R. Damasevicius (Eds.): ICIST 2016, CCIS 639, pp. 695–708, 2016.
DOI: 10.1007/978-3-319-46254-7_56

This means executing on the server-side the needed checks for the request at hand, finding the available vehicles, planning the appropriate route for the vehicle, and alerting the requesting users, vehicles and passengers for the adaptation. Of course, route changes will take place only when disruption is minimal, i.e. the other planned stops are satisfied as well as the timing constraints. In a smart environment, such a workflow takes advantage of cloud computing resources and an appropriate processing engine (Erl 2008; Wozniak et al. 2015).

By resorting to cloud-computing, transparent access e.g. to shared services, hardware and data is made possible, thus enabling a high level of *availability* and possibly performances. However, a cloud-based infrastructure can bring about delays for the virtual machine (VM) to be started, services to be allocated, etc. Primary goals for enterprises handling the said urban scenario of on-demand vehicles include: (i) having the minimum amount of disruption to paths of running vehicles, (ii) providing scalability to handle a variable number of requests with high availability, and (iii) minimise the potential delays.

The rest of this paper is structured as follows. Section 2 describes the smart city scenario that we have tackled. Section 3 describes the proposed software components that automatically provide workflow management, and enhance services with monitoring and dependability. Section 4 introduces our strategy for the prediction of transport needs. Section 5 discusses the related work, and Sect. 6 draws our conclusions.

2 Analysed Smart City Transportation Scenario

The proposed novel transport service would be an enhanced bus service, for which seats can be reserved, and the a priori established course can be slightly changed according to user requests and to other configurable settings. Such a bus service would allow citizens to cooperate with the institutions to make the urban transportation properly function, hence users would need to communicate their position and the desired destination for the bus service, as well as the useful timeframe for pick up. The services on a cloud computing system will then reply with a travel solution that satisfies user requests, indicating the best *rendezvous point* for the user to catch the bus.

In our model, a rendezvous point is a possible bus stop, one among many known by the transport operator, that is activated *on-demand* according to actual user requests, hence avoiding the need to serve bus stops when not strictly required, optimising vehicles travel cycles, and reducing energy consumption and CO2 emissions.

Figure 1 shows three possible routes connecting the Itako and Bando cities of the Ibaraki Prefecture, in the Japanese region of Kanto. By sensing actual user requests, the transport operator can determine in advance which of the six potential rendezvous points depicted (two for Bando and one for each of the other cities) will be served, hence planning the course to follow. User requests are gathered before the bus starts going and during its journey, then a server-side component would notify the bus drivers when it would be possible for the bus to change the route in order to serve requested rendezvous points.

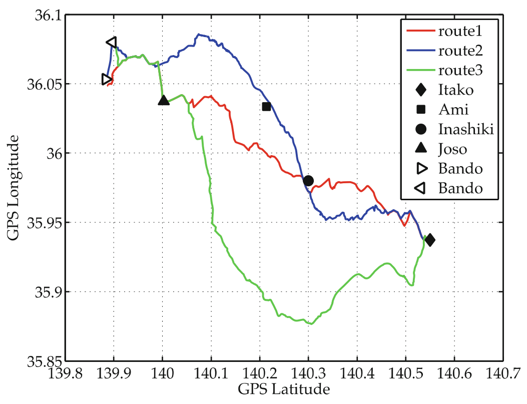

Fig. 1. Adapting routes in a transportation scenario

Choosing the right rendezvous points to satisfy user requests is an important concern: the server-side planning component has to properly manage the trade off between selecting the one nearest to the requesting user and modifying the established bus schedule (even on-the-fly). In Sect. 4 we will see how this schedule management can be assisted by an intelligent system able to predict future transport service requests in a given location.

The proposed solution goes beyond the urban transport scenario, i.e. other smart facilities can take advantage of the said data. The number of citizens wishing to use a transport service would be known for many locations while time goes on during the day. Then, a possible strategy, which can benefit mobile service providers, such as telco operators, is to take advantage of data representing user locations to estimate the mobile cell occupancy and operate needed counteractions.

Figure 2 shows an example of a simplified workflow, dubbed urban on-demand transportation, whereby citizens are able to actively request means of transportation to pass by a suggested place, in a given time-frame, as needed. On the server-side, once a user request has been received, a workflow is available to execute several services (activities), each corresponding to a step that has to be performed. In our reference model for the software system assisting such steps we will have one or more client applications enabling the user to submit a request, and wait for a reply. Hence, e.g. the ask transport vehicle step could be performed using a dedicated smartphone app that lets the user send detailed data such as: GPS position, desired time-frame for pick up, number of passengers.

Services on the server side are processes running, or started, according to the indication given by the workflow description, hence e.g. receive requests is the first step of an ad-hoc workflow, and is a process listening for incoming requests, residing inside a persistent web service, and compute travel solutions is a process started as a second step of the workflow once the previous step has been performed, etc. Since each service needed for a workflow completion in general can have its own preconditions, data and processing requirements, then each service should be handled ad-hoc to provide a proper quality of service. Let us suppose that compute travel solutions is a

CPU-bound process whose execution time has to be guaranteed, because, e.g. it could involve several scheduling decisions and transport combinations, starting from the temporal and location request given by users, aiming to provide the best solution for the users while reducing schedule changes. Instead, another service could simply provide immutable stored documents. Then, handling requests that trigger service execution requires the provisioning of ad-hoc computing resources to ensure the quality of service.

In a wider smart city scenario, we can consider the transport concern as one of the possible services that a smart city environment can offer, ranging from a consumption-aware city lighting management to the coordination of rescue interventions in case of urban issues. Therefore, to properly support the execution and integration of different smart city processes a proper infrastructure is needed.

3 Workflow Management System

A smart city process can be formalised by means of a workflow description and given to a workflow engine that timely starts the composing services in the desired order (Erl 2008; Polap et al. 2015) over a cloud infrastructure. Deploying and executing workflows on a cloud calls for a software architecture providing support for deploying, executing, and monitoring services, while minimising resource waste and standard delays affecting operations typical of this infrastructure (i.e. VM instantiations).

In the scenario depicted above (see Fig. 2), each workflow activity needs a dedicated description to allow proper resource allocations inside the cloud. Table 1 provides each activity with such additional data.

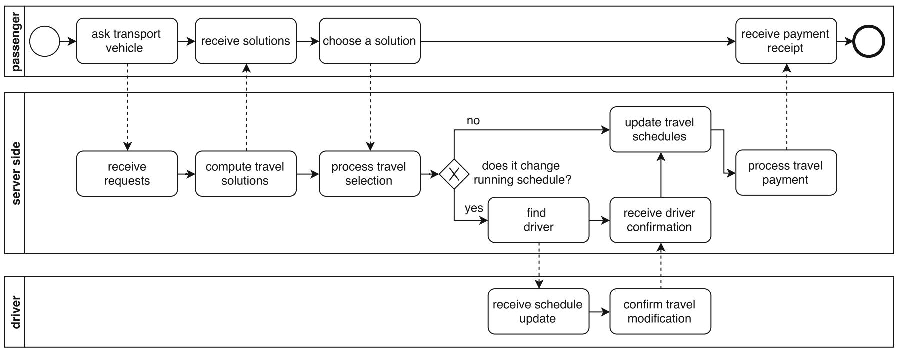

Fig. 2. An example of workflow urban on-demand transportation on a distributed system

In our solution, new workflow instances are started by a request coming from a `Web Service` (see the following section) and then a `Workflow Scheduler` finds the related workflow description and prepares resources for the execution of the first workflow activity, e.g. compute travel solutions, starting a dedicated VM with large hardware *flavor* configuration, where a flavour is a set of hardware characteristics for a

Table 1. Characterisation of server side activities for the urban on-demand transportation workflow for cloud resource management

Activity name	Start, end status	Resource type
Receive requests	On, on	No cloud
Compute travel solutions	Off, off	Dedicated, large VM
Process travel selection	Off, off	Dedicated, medium VM
Find driver	Standby, standby	Shared, small VM
Receive driver confirm	Standby, standby	Shared, small VM
Update travel schedules	Standby, standby	Shared, medium VM
Process travel payment	Off, off	Dedicated, medium VM

VM (Jackson 2012). Table 1 shows that each activity is characterised by a start and an end status, together with a description of the cloud resource type: the start status of a service environment (VM), such as already active, waiting for new process submissions (on); to be created and then turned on (off); in a standby state, reducing resource consumption and response time by avoiding the creation of a VM. If we specify that dedicated resources are needed, we will assign a separate VM for task execution, while for shared resources (useful e.g. to handle simple tasks like querying or updating a database or retrieving a document from the object storage) we run a process inside a shared VM. Web applications waiting for user requests (receive requests), and then triggering an actual workflow instantiation are not executed on a cloud.

3.1 Software Infrastructure

In order to support the features outlined above, we propose appropriate software components that enhance services and provide: (i) monitoring of workflow services, (ii) resource management, (iii) high availability.

Figure 3 shows how some of our components cooperate. `WorkflowScheduler` collects the user requests, determining whether to promptly accept them or not, according to the infrastructure load state. As any other scheduler, it manages a list of pending requests, using priority policies to determine their order, and interacts with other three components: `User Manager`, `Resource Manager` and `Workflow Repository`.

`User Manager` is responsible for the *AAA* service (Authentication, Authorisation, Accounting) related to user requests and will be asked to check user privileges and roles. Together with the number of requests previously accepted and completed for this user, it determines whether to lower the request priority. `Resource Manager` provides a high level representation of cloud resource states, holds the scheduling outcomes, and let us know whether a given resource is available in a certain time frame. `Workflow Manager` handles the descriptions of the workflow recorded inside the `Workflow Repository`, and is responsible for workflow instantiation and the monitoring of the completion of each inner activity for the requested workflow. It interacts with the `Cloud Manager` to ask for the execution on the cloud of the services implementing each activity.

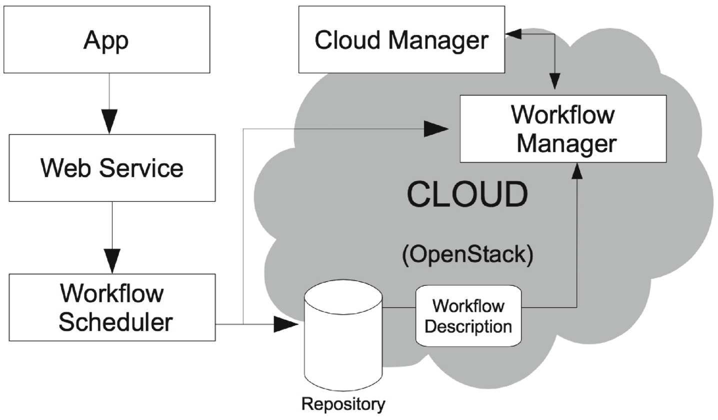

Fig. 3. Client requests served by a web server and a cloud-server

Cloud Manager is a *Facade* for cloud resources, hence providing the means for executing services inside the cloud infrastructure, which is actually managed by a cloud middleware, such as e.g. OpenStack[1]. A `Monitoring Service` gathers data on the completion time of each activity, then accounting information associated with the requesting user will be updated.

3.2 Flow of Operations and Use Case

Figure 3 shows the interactions between the components of the workflow management system for the completion of a workflow execution request. Using, for example, a web interface, the user can submit (either explicitly or not) an instantiation request of a specified workflow registered inside the management system.

This request arrives to `Workflow Scheduler`, that using the information coming from `User Manager` and `Resource Manager` determines whether to accept or reject the workflow execution request. It will then ask the `Workflow Repository` for the description of the requested workflow, providing the required services, resources and the estimated completion time for each of the inner activity. `Workflow Manager` will actually receive the instantiation request of a specified workflow, and starting from the first activity it will be responsible for monitoring the execution state of each activity within the workflow. `Workflow Manager` interacts with the `Cloud Manager` to request the instantiation of the cloud services needed by the current activity, according to the workflow description. We *contextualize* the new VM by loading a text file containing the activity information to which the VM is related to. The most important information required is the *activityID*, that the VM uses to notify the activity completion to `Workflow Manager`, as depicted in Fig. 3. In this way, `Workflow Manager` is informed on the termination of a specific activity of the

[1] www.openstack.org.

workflow, and can proceed with the destruction of the dedicated VM (because end state = off), asking also `Cloud Manager` to delete it. The `Workflow Manager` can then pass to the next activity described inside the workflow, and so on, until the workflow ends.

4 The WRNN Transport Request Predictor

The proposed management system takes care of the smart urban transport services in terms of number of passengers requesting the service in a certain location or rendezvous point in order to perfect seats allocation strategies and bus routes. In this section we present a module, called `Requests Predictor`, aiming at predicting the future load of the proposed services in terms of *number of passenger requests in a certain location*. Therefore, the number of requests to be serviced in the future can be easily computed. Prediction is based on the time series of requests, that can be observed and collected by the said management system during workflow execution. By considering the future expected load a fine-tuned management of human and transport resources can be achieved, hence avoiding both over provisioning and overloading. Without a prediction system, resources are usually managed by considering average values which can be computed over wide-ranging sets, hence are not appropriate for ensuring high quality of the service (Wozniak et al. 2016; Nowicki et al. 2016).

Generally, the benefits of such strategies decrease with the load increase. Moreover, when the amount of passengers overcomes available vehicles, passengers remain unserviced or suffer delays. However, since the amount of required resources (e.g. drivers, vehicles, rendezvous points, etc.) is unknown in advance, this often causes over-provisioning, with negative effects on management and the related cost.

Our experience in the field of neural networks points us towards a novel neural architecture called Wavelet Recurrent Neural Network (WRNN) (Napoli et al. 2014; Napoli et al. 2015). This architecture makes use of the wavelet analysis of a recurrent neural network topology. It has been shown that WRNNs are able to both predict the future evolution of a time series and perform the inverse wavelet transform in a similar way with respect to the Second Generation Wavelet Transform. A WRNN is the basis of our proposed Request Predictor (Napoli et al. 2016). This component predicts over time the amount of incoming passengers and therefore the service workload. The predictions are then used by the management system to allocate vehicles, drivers or other resources.

The predictor is used to model the number of incoming service requests in terms of number of passengers in a certain rendezvous point, and on the other hand it can be used to estimate when the transport will be available to service new passengers. Moreover, the module here presented is able to specialise a neural network to make predictions related to each rendezvous point (see Fig. 4). By specialising a neural network for each point it is possible to obtain a set of predictions that precisely map the status of the smart city-oriented transport services. Doing so it is then possible to provide the system manager with an early alert regarding the occupancy and urgency of several transport lines in order to better schedule and plan the service, allocate new

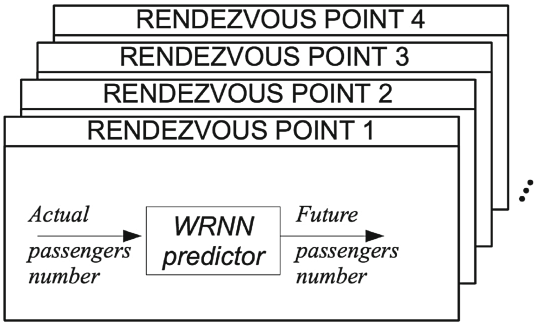

Fig. 4. The adopted WRNN predictor models and predicts the future trends regarding the number of passenger waiting on a rendezvous point. A predictor is specialised for one rendezvous point, hence a set of predictors is used.

vehicles, recall available drivers in service, as well as free resources when unneeded. The WRNN predictor operates with the time series indicating the passenger number at each rendezvous point over time. Firstly, the WRNN predictor transforms the time series in the wavelet domain.

The wavelet transform permits us to reduce data redundancies and obtain a representation that can express the intrinsic structure far more precisely than traditional analysis methods (e.g. Fourier transform). While the wavelet analysis exposes the time-frequency signature of the time series on different scales, the WRNN topology is the perfect complement to model the complexity of non-linear data correlation and perform data prediction on different scales. Thereby, a relatively accurate forecast of the passenger number can be achieved even when load peaks arise. The estimated result is fundamental for a management service that performs human and mechanical resources pre-allocation. The precision of our estimates allows just the right amount of resources to be used avoiding over provisioning. More details on WRNN architecture can be found in (Bonanno et al. 2012; Capizzi et al. 2012; Napoli et al. 2016).

In this paper, we have adopted the Biorthogonal wavelet decomposition (this kind of wavelet family is fully described in (Mallat 2009)), for which symmetrical decomposition and exact reconstruction are possible with finite impulse response (FIR) filters (Rabiner and Gold 1975). The transformed data are fed to the WRNN (see (Williams 1989; Mandic and Chambers 2001)).

The proposed WRNN consists of a Recurrent Neural Network architecture an input layer of 7 neurons, two hidden layers of 8 neurons with a radial basis activation function, a linear output layer with one linear neuron, and two delayed input units as well as two delayed feedback units from the output (see Fig. 5). The neural network is fed with the data constituted by time steps of a time series in the wavelet domain (Napoli et al. 2010; Bonanno et al. 2014a, b) representing the number of passengers requesting a transport services in a given location. Using a discrete time index τ we can call $q^{\mu}(\tau)$ the number of passengers at a time τ for a certain rendezvous point μ. By applying the wavelet transform to the time series $q^{\mu}(\tau)$ we obtain the related representation in the wavelet space.

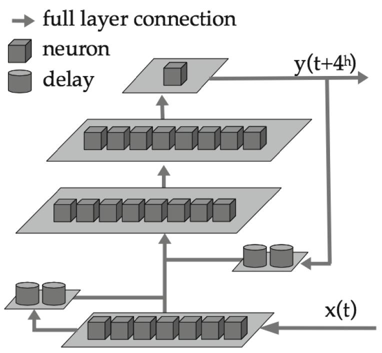

Fig. 5. Devised neural network. Delays and feedback are obtained by using the relative delay lines and operators (D).

$$q^{\mu}(\tau) \xrightarrow{\hat{W}} \left[q^{\mu}_{a_6}(\tau), q^{\mu}_{d_6}(\tau), q^{\mu}_{d_5}(\tau), q^{\mu}_{d_4}(\tau), q^{\mu}_{d_3}(\tau), q^{\mu}_{d_2}(\tau), q^{\mu}_{d_1}(\tau)\right] \tag{1}$$

Since we have used a 5 level transform, we have defined $q^{\mu}(\tau)$ as shown in Eq. 1, where the arrow represent the transform operation, $\hat{W}$ represents the biorthogonal wavelet decomposition and the resulting vectors have the component values on the different decomposition scales (from scale 1 to scale 6, where the letter d indicates the wavelet coefficients and a_6 the residuals on the most gross scale).

For reasons related to the noise signature we were not interested in the last component of the transformed series, therefore we had an input vector $x^{\mu}(\tau)$ in the form shown in Eq. 2.

$$x^{\mu}(\tau) = \left[q^{\mu}_{a_6}(\tau), q^{\mu}_{d_6}(\tau), q^{\mu}_{d_5}(\tau), q^{\mu}_{d_4}(\tau), q^{\mu}_{d_3}(\tau), q^{\mu}_{d_2}(\tau), q^{\mu}_{d_1}(\tau)\right] \tag{2}$$

The overall input set, considering N time steps, can be then represented as a $N \times 7$ matrix where the i-th row represents the i-th time step. Each row of this dataset is given as input value to the 7 input neurons of the proposed WRNN. The properties of this network make it possible, starting from an input at a time step τ_n, to predict the number of requests and throughput at a time step $\tau_n + \sigma$.

$$\tilde{x}^{\mu}(\tau_{n+\sigma}) = \hat{N}_{\mu}[x^{\mu}(\tau_n)] \tag{3}$$

In this way, the WRNN acts like a function $\hat{N}_{\mu}$ as defined by Eq. 3, where σ is the number of time steps of forecast in the future, and the tilde on the symbol $\hat{x}$ indicates that it is a prediction instead of a measurement. The number σ is not specified in the equation since it depends on the sampling frequency of the input and output time series. In our work, the data had a sampling period of 10 min, while we predicted the number of passengers 4 h ahead, therefore with $\sigma = 24$. To model the time series and then to predict their future evolution the neural network is firstly trained with the historical time series, several training epochs are interleaved with the related supervised pruning procedure. When the process in concluded, a network starts to provide forecasts related to a specific service for which it was trained. By collecting each service forecast we obtain a complete map $\tilde{S}(\tau_{n+24})$, predicted beforehand, as defined by Eq. 4.

$$\tilde{S}(\tau_{n+24}) = \{\tilde{x}^{\mu}(\tau_{n+24})\forall\mu\}. \tag{4}$$

The results of the WRNN predictor are shown in Fig. 6. The predictor was able to propose an early estimate of the future number of passengers requesting a transport services in a certain rendezvous point. The maximum error occurring in the prediction is two passengers with respect to the effectively measured number. In our experiments the WRNN predictor has been used in order to schedule transportation means and manage an optimized planning of the related human and mechanical resources, and such operations are made possible thanks to predicted number of passengers, due to their availability in advance. Moreover, since each rendezvous point has been associated to a WRNN predictor, the proposed solution is general for any number and kind of transport services as well as for integrated transportation services. Finally due to the cloud technology, the system could be expanded and scaled on demand.

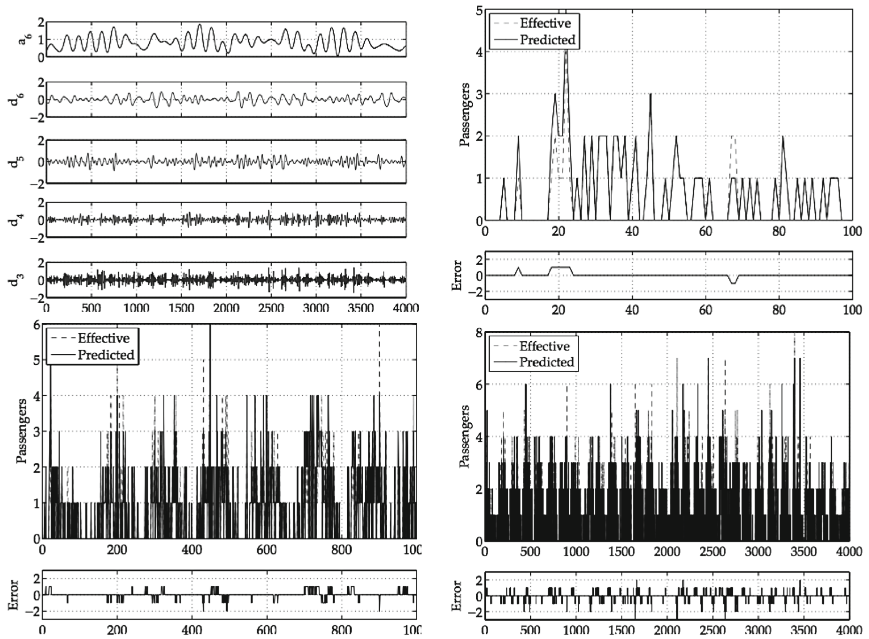

Fig. 6. From left to right and top to bottom: the most gross scales of the wavelet decomposition of the historical time series of passengers for a selected rendezvous point; the predicted and measured throughput, the errors on the throughput predictions, the predicted and measured number of requests, the errors on the requests predictions. The gray dashed lines represent the effective measurements, the overlapping black lines represent the predictions made by the WRNN predictors.

5 Related Works

Future urban environments will gather data from several cheap sensors, using cross-correlation and analytical models to mine valuable and reliable information from a lot of noisy and unreliable heterogeneous sensors. In (Puliafito et al. 2015) authors propose a hierarchical architecture to cope with the heterogeneous nature of sensed data in a World Wide Sensor Network scenario, using a two level data abstraction to uniform exposed information. While this work is focused on how data will have to be collected and organised, our work describes how to drive Smart City processes, that use sensed data, on a Cloud infrastructure, specifically to enhance the transportation service.

Other contributions have covered the interaction between sensors and Cloud services. In (Fazio et al. 2012) authors propose an architecture to assist the integration between IoT devices and federated Cloud providers, focussing on the security concerns that this integration involves, providing an IoT Cloud-based architecture that is able to support self-identification and secure data transfer from self-configuring devices to Cloud providers.

In (Liebig et al. 2016) the authors present a system for individual trip planning that incorporates the current traffic state, gathered by a sensor network positioned in the city environment, and future traffic condition, that is provided by a spatio-temporal random field model leveraging sensed data. They also use a Gaussian process regression to estimate traffic flow in areas with low sensor coverage. This approach not only differs from ours in the model they used to achieve predictions, but also in the motivations and the service point of view. In their approach it is the final user that will adapt his/her trip plan to the one suggested by the proposed system. In our approach it is instead the city environment, i.e. the devised city transportation service, that adapts itself to user requests. Therefore, where they have streets and city points that users have to avoid, according to current and future traffic conditions, we have rendezvous-points, i.e. city locations, that the transportation service vehicles have to reach to serve user requests. On the other hand, our approach can be supported by the service they proposed to make the bus drivers able to choose, among different possibilities, the fastest trip to reach the next rendezvous point to be served.

In (Zhu et al. 2016) the authors propose a public vehicle (PV) system that uses a cloud infrastructure to devise scheduling strategies and paths for PVs based on the demands of passengers. PVs are electric vehicles that can transport more passengers than a taxi and do not have stops or routes like a bus service. The service they propose is different from ours, since we have predefined bus stops (rendezvous-points) which are activated on demand according to user requests, hence adapting the overall trip schedule for the transportation vehicles. Furthermore, in contrast to our approach, they do not make use of a prediction component to assist their scheduling decisions, relying instead on a set of constraint based algorithms to compute the best solution to satisfy user requests.

In (Guitart et al. 2009) an extensive survey of the literature on quality of service, availability and performance for distributed applications has been given. All the approaches in (Guitart et al. 2009) are not intended to be used in combination with cloud resources. Hence, further support is needed as shown in the sections above.

In (Zhang and Qi 2005) the authors use pre-processing stages in order to feed filtered data to neural networks to model time series with both seasonal and trend patterns. Hybrid models are widely used in the literature in order to model phenomena and obtain forecasting software systems for a wide range of purposes, such as e.g. hydro-geological time series and the related risk assessment (Jain and Kumar 2007; Lohani et al. 2012). Other kinds of neural network related approaches have been developed for traffic prediction, e.g. basing on a flexible neural tree and particle swarm optimisation algorithm (Chen et al. 2012).

The novelty of the presented work lies on the pre-processing of the data that, in this system, are transformed to the wavelet domain before they are given to the neural network. Moreover, the neural network itself is able to perform the inverse transform: that is a non trivial feature since it permits us to encapsulate it in a module that can be embedded in more complex systems. In this way the neural architecture is not anymore a stand-alone and separated component, but can be integrated in a software system to internally cooperate with the other components.

6 Conclusions

This paper has proposed a software architecture that gives support when having to deploy, execute, and monitor services that execute on a cloud and according to workflows. A smart city scenario has been put forward for enhanced intelligent services that are related to public transport and can gather useful data to improve the communication infrastructure. The server-side computing architecture, while being independent of specific workflows, has shown to be flexible enough to handle mobile user requests and properly govern the life-time of services executing on a cloud. Moreover, our suggested components realise all the interconnection work needed to let a workflow execute on a cloud infrastructure.

In our solution, a component has been specifically devised to plan the needed transport service ahead of time by modelling incoming requests and analysing them in order to remove noise, while characterising repetitive trends. Then, we can start operations, such as planning vehicles routes and driver shifts, avoiding over-provisioning.

References

Batty, M., Axhausen, K.W., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., Portugali, Y.: Smart cities of the future. Eur. Phys. J. Spec. Top. **214**(1), 481–518 (2012)

Benevolo, C., Dameri, R.P., D'Auria, B.: Smart mobility in smart city. In: Torre, T., Braccini, A. M., Spinelli, R. (eds.) Empowering Organizations: Enabling Platforms and Artefacts. Lecture Notes in Information Systems and Organisation, vol. 11, pp. 13–28. Springer, Heidelberg (2016)

Bonanno, F., Capizzi, G., Sciuto, G.L., Napoli, C., Pappalardo, G., Tramontana, E.: A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in IGSs by using WRNN predictors and GPU parallel solutions. In: Proceedings of IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), Ischia, pp. 1077–1084 (2014). doi:10.1109/SPEEDAM.2014.6872127

Bonanno, F., Capizzi, G., Napoli, C.: Some remarks on the application of RNN and PRNN for the charge-discharge simulation of advanced Lithium-ions battery energy storage. In: Proceedings of IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), Sorrento, pp. 941–945 (2012). doi:10.1109/SPEEDAM.2012.6264500

Bonanno, F., Capizzi, G., Coco, S., Napoli, C., Laudani, A., Sciuto, G.L.: Optimal thicknesses determination in a multilayer structure to improve the spp efficiency for photovoltaic devices by an hybrid FEM—cascade neural network based approach. In: Proceedings of IEEE International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM) (2014)

Capizzi, G., Napoli, C., Paternò, L.: An innovative hybrid neuro-wavelet method for reconstruction of missing data in astronomical photometric surveys. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part I. LNCS, vol. 7267, pp. 21–29. Springer, Heidelberg (2012)

Chen, Y., Yang, B., Meng, Q.: Small-time scale network traffic prediction based on flexible neural tree. Appl. Soft Comput. **12**(1), 274–279 (2012)

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J.R., Mellouli, S., Nahon, K., Pardo, T., Scholl, H.J., et al.: Understanding smart cities: an integrative framework. In: Proceedings of 45th Hawaii International Conference on System Science (HICSS), pp. 2289–2297. IEEE (2012)

Erl, T.: SOA Design Patterns. Pearson Education, Boston (2008)

Fazio, M., Paone, M., Puliafito, A., Villari, M.: Heterogeneous sensors become homogeneous things in smart cities. In: Proceedings of International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 775–780. IEEE (2012)

Fornaia, A., Napoli, C., Pappalardo, G., Tramontana, E.: Using AOP neural networks to infer user behaviours and interests. In: Proceedings of the 16th Workshop From Objects to Agents (WOA), vol. 1382 (2015)

Guitart, J., Torres, J., Ayguadé, E.: A survey on performance management for internet applications. Concurrency Comput. Pract. Exp. **22**(1), 68–106 (2009)

Jackson, K.: OpenStack Cloud Computing Cookbook. Packt Publishing Ltd., Birmingham (2012)

Jain, A., Kumar, A.M.: Hybrid neural network models for hydrologic time series forecasting. Appl. Soft Comput. **7**(2), 585–592 (2007)

Liebig, T., Piatkowski, N., Bockermann, C., Morik, K.: Dynamic route planning with real-time traffic predictions. Inf. Syst. (2016, in press). doi:10.1016/j.is.2016.01.007, http://www.sciencedirect.com/science/article/pii/S0306437916000181

Lohani, A., Kumar, R., Singh, R.: Hydrological time series modeling: a comparison between adaptive neuro-fuzzy, neural network and autoregressive techniques. J. Hydrol. **442**, 23–35 (2012)

Mallat, S.: A Wavelet Tour of Signal Processing: the sparse way. Academic press, Burlington (2009)

Mandic, D.P., Chambers, J.: Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures And Stability. Wiley, New York (2001)

Napoli, C., Bonanno, F., Capizzi, G.: Exploiting solar wind time series correlation with magnetospheric response by using an hybrid neuro-wavelet approach. In: IAU Symposium 274, vol. 6, pp. 156–158. Cambridge University Press (2010). doi:10.1017/S1743921311006806

Napoli, C., Pappalardo, G., Tina, G.M., Tramontana, E.: Cooperative strategy for optimal management of smart grids by wavelet RNNs and cloud computing. IEEE Trans. Neural Netw. Learn. Syst. **27**(8), 1672–1685 (2015)

Napoli, C., Pappalardo, G., Tramontana, E.: Improving files availability for bittorrent using a diffusion model. In: 23rd IEEE International WETICE Conference, pp. 191–196 (2014). doi:10.1109/WETICE.2014.65

Napoli, C., Pappalardo, G., Tramontana, E.: A mathematical model for file fragment diffusion and a neural predictor to manage priority queues over bittorrent. Int. J. Appl. Math. Comput. Sci. **26**(1), 147–160 (2016)

Nowicki, R.K., Nowak, B., Wozniak, M.: Application of rough sets in k nearest neighbours algorithm for classification of incomplete samples. In: Kunifuji, S., Papadopoulos, G.A., Skulimowski, A.M.J., Kacprzyk, J. (eds.) Knowledge, Information and Creativity Support Systems. Advances in Intelligent Systems and Computing (AISC), vol. 416, pp. 243–257. Springer, Heidelberg (2016). doi:10.1007/978-3-319-27478-2_17. ISSN 2194-5357

Polap, D., Wozniak, M., Napoli, C., Tramontana, E.: Real-time cloud-based game management system via cuckoo search algorithm. Int. J. Electron. Telecommun. **61**(4), 333–338 (2015)

Puliafito, A., Celesti, A., Villari, M., Fazio, M.: Towards the integration between IoT and cloud computing: an approach for the secure self-configuration of embedded devices. Int. J. Distrib. Sensor Netw. **11**(12), 268–860 (2015)

Rabiner, L.R., Gold, B.: Theory and Application of Digital Signal Processing. Prentice-Hall, Englewood Cliffs (1975)

Williams, R.J.: A learning algorithm for continually running fully recurrent neural networks. Neural Comput. **1**, 270–280 (1989)

Wozniak, M., Marszalek, Z., Gabryel, M., Nowicki, R.K.: Preprocessing large data sets by the use of quick sort algorithm. In: Skulimowski, A.M.J., Kacprzyk, J. (eds.) Knowledge, Information and Creativity Support Systems: Recent Trends, Advances and Solutions. Advances in Intelligent Systems and Computing (AISC), vol. 364, pp. 111–121. Springer, Heidelberg (2016). doi:10.1007/978-3-319-19090-7_9. ISSN 2194-5357

Wozniak, M., Polap, D., Borowik, G., Napoli, C.: A first attempt to cloud-based user verification in distributed system. In: Proceedings of IEEE Asia-Pacific Conference on Computer Aided System Engineering (APCASE), pp. 226–231, July 2015

Zhang, G., Qi, M.: Neural network forecasting for seasonal and trend time series. Eur. J. Oper. Res. **160**(2), 501–514 (2005)

Zhu, M., Liu, X.-Y., Qiu, M., Shen, R., Shu, W., Wu, M.-Y.: Transfer problem in a cloud-based public vehicle system with sustainable discomfort. Mobile Netw. Appl., 1–11 (2016, in press). doi:10.1007/s11036-016-0675-y