DynNav: Toward Open and Interoperable Navigation Services

DynNav: Toward Open and Interoperable Dynamic Navigation Services

Piergiuseppe Bettassa, *Politecnico di Torino*, piergiuseppe.bettassa@polito.it

Cecilia Corbi, *Telecom Italia Lab*, ceciliamaria.corbi@telecomitalia.it

Giorgio Ghinamo, *Telecom Italia Lab*, giorgio.ghinamo@telecomitalia.it

Pietro Leggero, *Intelligent App*, p.leggero@mytaxi.net

Fulvio Risso, *Politecnico di Torino*, fulvio.risso@polito.it

Riccardo Sisto, *Politecnico di Torino*, riccardo.sisto@polito.it

Francesco Vadalà, *Telecom Italia Lab*, francesco.vadala@telecomitalia.it

(Headline)

So far navigation devices, including navigation apps for smartphones, have been proprietary and closed. A new scenario is emerging with the Open Mobile Alliance Dynamic Navigation Enabler, which lets developers create novel navigation services characterized by openness and interoperability across different information providers.

1 Introduction

Navigation Devices (NDs), which are common tools for getting driving assistance, are increasingly integrating local information (e.g., maps, user position) with remote data such as real-time traffic information. However, the ND of one manufacturer is hardly capable to access data provided by a different service provider, as current systems are either *proprietary*, or even totally *closed*.

Google Navigation and Apple Maps are well-known examples of proprietary solutions. They define a set of APIs, exploitable by other parties, but they do not guarantee interoperability with other solutions; even more importantly, those APIs may have some technical/legal limitations. For instance, Google does not allow the use of its data in applications other than Google Navigation [1], [2]. In addition, Google APIs are subject to change at any time. Other solutions, such as TomTom or Garmin either rely on the old radio-based broadcast communication channel, or exploit a closed architecture (services, protocols), hence preventing users from switching to another provider. Moreover, previous solutions, having monolithic nature, do not allow the use of alternative components (e.g., better real-time traffic information sources), and offer only limited chances to customize them.

In order to overcome many of these limitations, recently the Open Mobile Alliance (OMA) standardized an open protocol for dynamic road navigation services. The OMA Dynamic Navigation Enabler [3], in short, *DynNav*, introduces a bidirectional communication channel and a modular approach, while reusing existing standards for some specific features. The bidirectional channel allows users to ask for or receive only the information they are interested in, by filtering data based on routes, geographic areas and time [4]. The standardized modular approach allows multiple services to be created on top of the defined components and it enables the same service to be provided by multiple operators (avoiding users lock-in), and to be implemented in multiple flavors. In this way, new actors can participate in the provisioning of navigation services, with the creation of valuable and/or specialized components, such as route computation algorithms optimized for vertical markets (e.g., goods delivery), precise traffic information, and more. Modularity and openness allow users not only to exploit multiple providers in their solution (e.g., one for route computation, the other one for traffic information), but also to seamlessly change provider.

Among the possible players, telecom mobile operators may have huge benefits from an open standard for navigation services. First, they are constantly looking for the possibility to provide new services to their customers, and dynamic navigation is an appealing option for many users. Second, mobile operators can obtain traffic information by exploiting their own assets, in particular by (anonymously) tracking the position of their mobile users, hence providing real-time traffic information and re-routing capabilities based on real-time data.

In this context, and in particular in the case of smart cities, where multiple sources of real-time data are available, such as traffic, parking lots availability, and more, DynNav-enabled solutions can not only provide navigation services, but also become aggregators of multiple information sources, helping users to find localized information such as restaurants or attractions, possibly matching user's preferences and interests.

2 The DynNav solution

OMA, the leading industry forum for developing market-driven, interoperable mobile service enablers, completed the standardization process of DynNav in September 2012. This represents an additional step toward the full support of navigation applications by the OMA standardization framework [3].

2.1 Architecture

Figure 1 presents a possible architecture of a navigation service based on DynNav. The server is the middle block, while the left block represents a typical ND (e.g., a dedicated device or a smartphone). A DynNav client can also be an application residing on a server (right side in figure), such as a web-based journey planner.

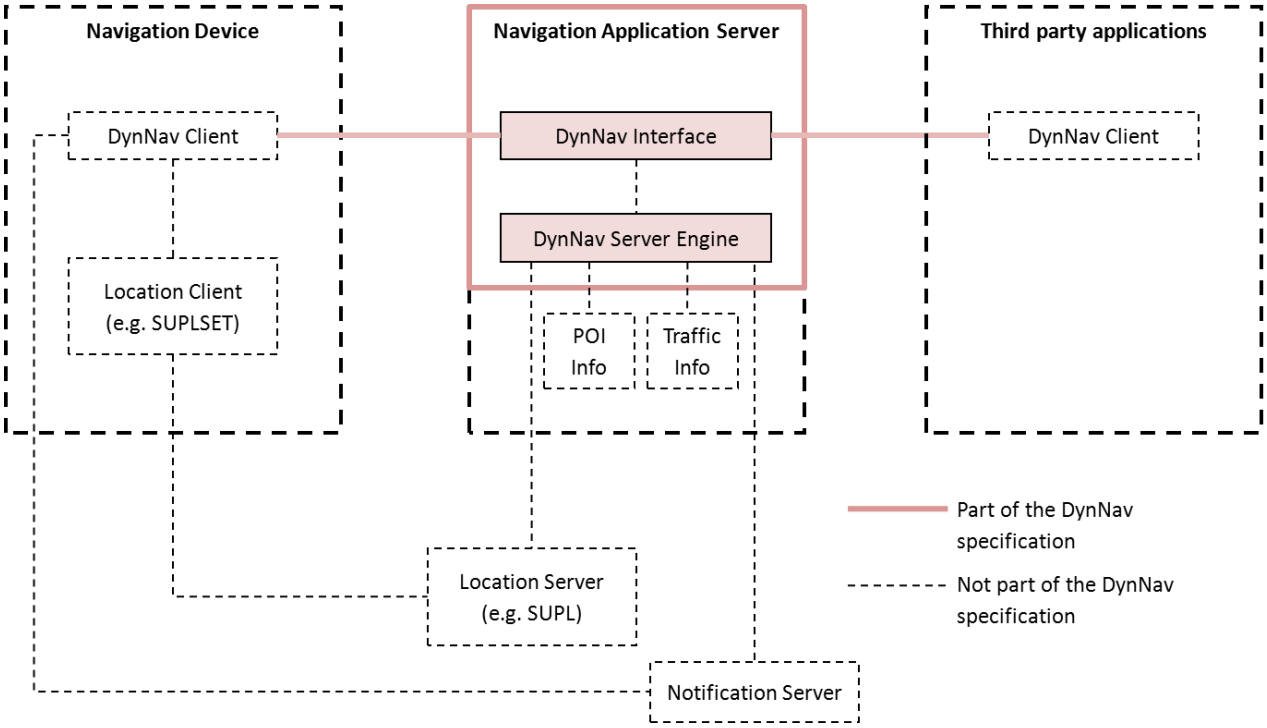


Figure 1: DynNav Enabler Architectural Diagram

DynNav specifies the interaction between a DynNav client and a DynNav server (the straight lines in Figure 1) while the interactions with the other components (the dashed lines) rely on protocols that are outside the scope of the DynNav specification. Additional elements, not depicted in Figure 1, can be introduced to optimize the bandwidth consumption, which is critical in mobile communications. For example, proxy servers can cache frequent responses.

2.2 Protocol

The DynNav protocol follows the OMA RESTful Network NetAPIs guidelines [5] and bases on the Representational State Transfer (REST) [6] paradigm.

A server that implements DynNav provides the following main functions:

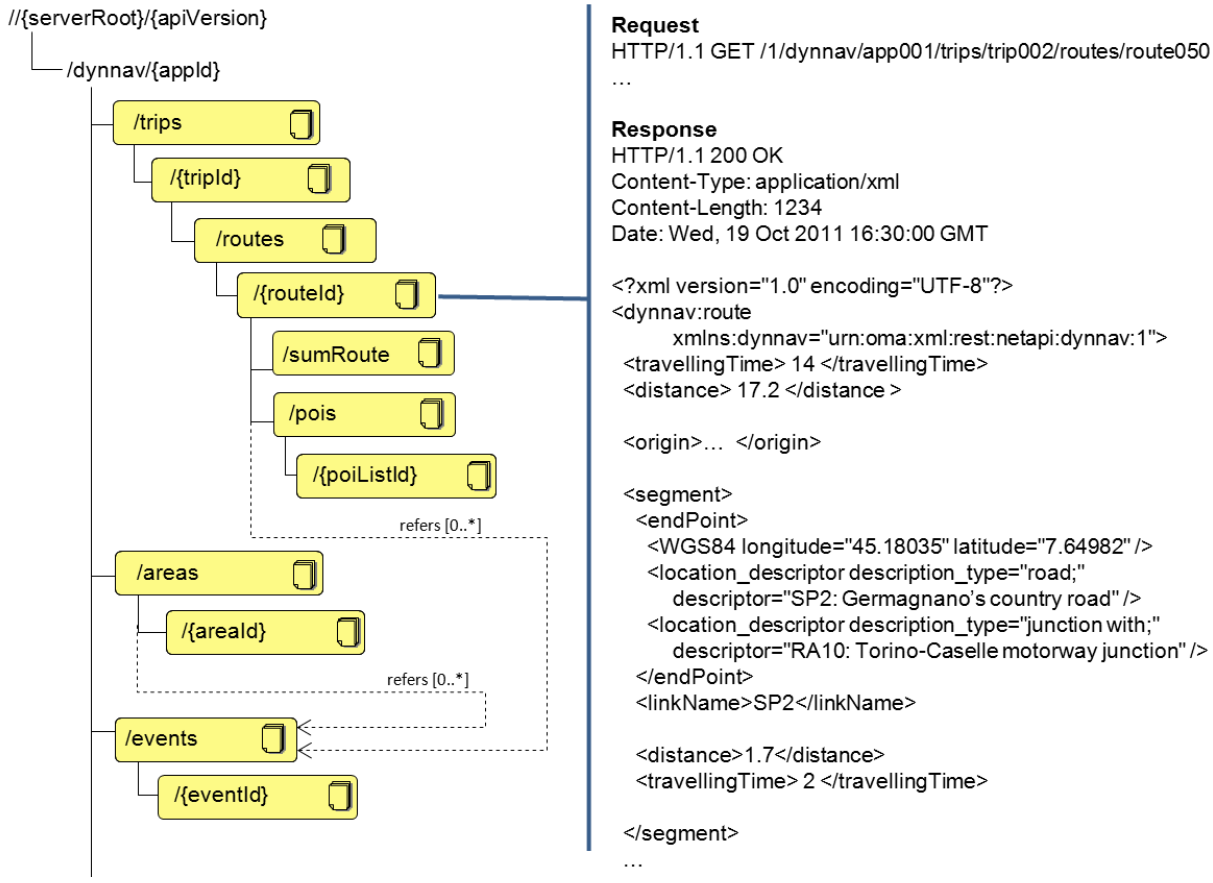
1. Analysis of client-defined trip parameters and proposal of a set of routes, based on real-time and forecast traffic data;
2. Provisioning of real-time and forecast traffic information related to a set of routes (or geographical areas), previously proposed by the DynNav server itself or by the ND;
3. Proposal of new routes when the quality indicators associated with the current route become unacceptable or the user deviates from the current route;
4. Provisioning of complementary information such as Points of Interest (POI) related to either a route or an area;
5. Possibility for the client to subscribe to a set of routes, and to be notified of available information about these routes only when the user is driving along one of them.

It is worth noting that a client could not be allowed to access all the functions listed above: the DynNav service provider can allow/deny access to some functions depending on the user contract or other criteria.

2.3 Data structures

The DynNav specification reuses, whenever possible, data structures already defined in previous standards, with the objective of speeding up the implementation and facilitating the integration of additional components/services into the framework. Particularly, it exploits the Transport Protocol Experts Group (TPEG) standards for road traffic messages and location, traffic events, road performance (e.g., travelling time) and location entity description [7]. Moreover, it exploits IETF RFC 4776 [8] for civic addresses and the W3C "Points of Interest Core" draft [9] for POI information. The drawback of this approach is the impossibility to obtain a fully optimized protocol, as data formats and protocol messages may be more verbose than necessary. As no public standards are currently available for encoding route information, a novel encoding schema has been proposed: a route is represented by a sequence of segments (i.e., road sections without intersections), each one including segment origin, destination, name, measured or forecast performance parameters (travelling time in regular conditions, delays, expected speed) and segment shape (sequence of points for its graphical representation on the map).

Figure 2 depicts a portion of the DynNav resource tree (the complete structure is shown in [4]) and a sample of request and response messages. The curly brackets identify parametric parts of the URIs. The *trips* resource contains a collection of trip resources, each one identified by its unique *tripId*. A trip resource includes information about the journey (e.g., source and destination) along with nested resources representing alternative routes to the destination. These resources, identified by their unique *routeId*, include routing information and may also include references to sets of events, grouped by categories. Users can obtain the details by sending a request to the server, containing the list of events to be retrieved. The selection of interesting events can either be done manually or automatically by the client application. This choice can reduce the amount of bytes transmitted over the network because only interesting events are retrieved, even if it introduces a small delay due to the additional request. If the same event is shared by multiple resources, such as different routes or areas, it is transmitted only once. Events can be stored in proxy servers, if present, reducing the number of requests to the main server.


//{serverRoot}/{apiVersion}
/dynnav/{appId}
/trips
/{tripId}
/routes
/{routeId}
/sumRoute
/pois
/{poiListId}
refers [0..*]
/areas
/{areaId}
refers [0..*]
/events
/{eventId}
Request
HTTP/1.1 GET /1/dynnav/app001/trips/trip002/routes/route050
…
Response
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1234
Date: Wed, 19 Oct 2011 16:30:00 GMT
<?xml version="1.0" encoding="UTF-8"?>
<dynnav:route
xmlns:dynnav="urn:oma:xml:rest:netapi:dynnav:1">
<travellingTime> 14 </travellingTime>
<distance> 17.2 </distance >
<origin>… </origin>

<endPoint>
<WGS84 longitude="45.18035" latitude="7.64982" />
<location_descriptor description_type="road;"
descriptor="SP2: Germagnano's country road" />
<location_descriptor description_type="junction with;"
descriptor="RA10: Torino-Caselle motorway junction" />
</endPoint>
<linkName>SP2</linkName>
<distance>1.7</distance>
<travellingTime> 2 </travellingTime>

…

Figure 2: Excerpt of resource tree (left) and of the messages for reading a route resource (right)

2.4 Use case example

A typical application of the DynNav service refers to a ND without path computation capabilities, which may originate the message flow depicted in Figure 3 (a detailed description is available in [4]). In this scenario, the user first specifies the parameters of a trip by sending Message #1, which triggers the creation of a new trip resource in the server, including all the details about the trip as provided by the user. Trip creation also forces the server to calculate a set of routes that satisfy the constraints.

Then the client retrieves and selects one of the possible routes proposed by the server by Messages #2 and #3. The route includes links to the occurred events (whose descriptions are stored on the server) and a field that specifies the category (e.g., traffic, weather) of each event. A client application may decide not to retrieve all those events from the server (Message #4), based on user preferences or other criteria (e.g., priorities). Message #5 creates a subscription to the notification service in order to receive real-time traffic information updates and the proposal of alternative routes. The DynNav server will use user-provided location data in order to update the user's status and send notifications about traffic and other events specified in the user's subscription. If the estimated travelling time becomes too high, e.g., because of road congestion, a new route can be suggested. Message #6 represents a notification from the server and it contains the references to related events, which can be retrieved as in Message #4.

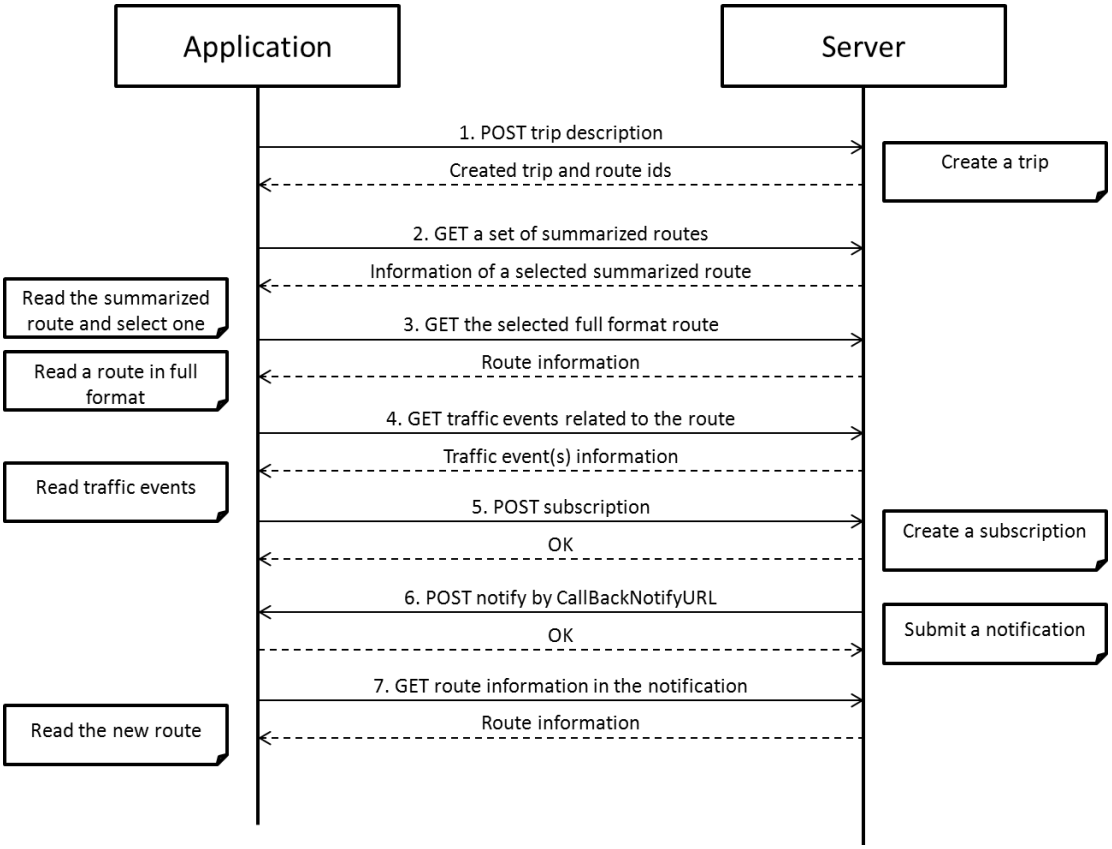


Figure 3: A possible DynNav message flow

2.5 Deployment scenarios

The flexibility of the DynNav standard enables the creation of rich navigation services without locking users with a specific service provider and/or using a fixed set of functions. This is possible because the same service (e.g., route computation) can be provided by different entities, and the basic set of DynNav messages can be combined in (almost) arbitrary ways to create complex applications. For example, DynNav supports both smart NDs, which have the capability to calculate routes and rely on central services only for real-time information (e.g., traffic), and lightweight NDs, which delegate everything but the user interaction (e.g., route display) to central services.

The possibility to exploit services provided by multiple entities also enables the creation of applications targeting very specific vertical markets. For example, a shipping goods company can opt for a lightweight ND for its employees, relying on an in-house service for route computation, with the ability to minimize delivery costs, and on a telecom operator to obtain real-time traffic information. Along the same line, a company can setup only a specialized service that suggests the best points of interest based on the user's preferences or other parameters, relying on the fact that other providers can offer the other information required for building a whole navigation service.

3 Validation

We implemented the DynNav standard in a prototype that includes both the client and server portions of the specification. We used those components to carry out an extended set of tests aimed to validate the characteristics of the standard in a real world environment and to give an insight of the performance achievable by this solution. Particularly, we were interested in checking whether DynNav is suitable for delivering navigation services to real mobile devices. Although our prototype was not engineered to compete with commercially available solutions, we compared DynNav with the widely used Google Directions Web Services [1], which offers similar primitives and exploits a similar data structure.

3.1 Prototype

The DynNav server was developed in Java and installed on a GlassFish application server, while the ND hosting our application was a low-end Android 2.2 smartphone clocked at 768MHz with 512MB of memory.

Some specific functions of the server were set up by reusing existing external components: route calculation is delegated to an external web service (http://openrouteserver.org), while traffic data are simulated using a real data source to feed the simulator.

The DynNav prototype described here has been shown in an OMA demo session. Details (presentation and interview) can be found in [10], [11].

3.2 Tests

We focused on a common worst-case usage pattern in which a lightweight DynNav client asks for the entire information related to a trip, delegating route computation to the server. This implies that the client device is forced to request even the data necessary to display the chosen path on the map.

First, we measured the amount of data generated by the protocol, which, if excessive, could have a negative impact on performance in low-bandwidth environments. The overall results, computed by averaging several trips of different complexities, showed that DynNav messages are in average 40% bigger than Google messages. However, this overhead also includes information about real-time driving times and events (e.g., traffic jams, accidents) associated with the route, not reported by Google Directions. In order to limit this overhead, we enabled the HTTP compression that is supported by most HTTP implementations and reduces the total bytes transferred. In our experiments, we observed a compression rate of about 2.8:1 in route messages, with a 10% processing time increase (due to compression/decompression) in the ND. With compression enabled, DynNav messages are in average 35% bigger than (compressed) Google messages.

In the second test we measured the time elapsed between starting a route request and receiving the associated response. This time needs to be as short as possible as it represents the waiting time for the ND user. Tests showed that the average latency for obtaining the route is approximately 300ms, excluding the time required by the server to compute the route, calculated on 25 realistic trips of different complexity. While this number is three times the one experienced in Google Directions, this value is still reasonable as human beings will hardly notice the difference in terms of responsiveness between the two systems. The higher latency is due to the larger size and number of messages (i.e., HTTP GET requests) needed to complete the same operation. This is required in order to enable a greater extent of flexibility in the protocol and support different deployment scenarios with a limited set of primitives, such as NDs with/without route computation capabilities and/or navigation maps.

A third test evaluated the processing overhead of the protocol in the ND, with the final objective of assessing the possibility to execute DynNav services on low-end user terminals. Our measures showed that the average processing cost for a message transporting route information in the selected ND is about 2.5ms for each Km of the route, which represents an acceptable value even for long routes. This confirms that the choice of using rich XML messages, which are known to be more computationally demanding than binary encoded messages, does not represent an issue on modern user terminals at least in this use case.

Table 1 shows how the performance of the system changes when dealing with routes of increasing lengths, with HTTP compression. These data confirm that the DynNav performance figures remain acceptable within the typical route complexity range.

	Message size [KB]		Transmission Time [ms]		Processing time on client [ms]
Route length [kilometers]	Google Directions	DynNav	Google Directions	DynNav	
53,73	4,29	5,24	72,15	258,85	160,32
244,84	7,18	9,34	76,30	263,45	719,93
600,96	20,63	31,48	95,62	295,76	1760,15
878,95	34,07	45,53	114,88	312,98	2577,24
1034,00	29,40	39,43	108,19	305,12	3030,57

Table 1: Tests results

4 Conclusions

Based on the new scenarios enabled by DynNav and on our experiments, we can conclude that DynNav has the potential to change the way navigation services are conceived, implemented and deployed, by making it more open and interoperable. Improvements are also possible and OMA is working on an enhanced version. Future work should be directed to optimize the definition of resources and to add new resources such as detailed parking information, public transportation, indoor navigation, weather conditions. In fact, the warm acceptance of this standard among different players (ND/smartphone manufacturers, telecom operators) is pushing for further evolutions, particularly with respect to value-added services (e.g., points of interest, support for vertical applications such as logistics). Additional studies, in collaboration with other mobile operators, are currently ongoing to define optimized mechanisms for reducing and compressing the amount of transferred data.

The prototype that has been developed to validate the DynNav solution shows excellent results, even on a non-optimized implementation. Although in some cases the performance looks inferior than in proprietary solutions (e.g., Google Directions) albeit hardly noticeably by final users, our DynNav prototype enables interoperability and greater flexibility thanks to its additional features, such as push-based notification services and customizable real-time information.

5 Biographies

Piergiuseppe Bettassa graduated in Computer Engineering from Politecnico di Torino in 2011 and he enrolled in the PhD program. His research interests are on formal methods applied to security protocols and security-aware applications, software engineering and model driven development. Contact him at piergiuseppe.bettassa@polito.it.

Fulvio Risso, PhD, is assistant professor with the Control and Computer Engineering Department of Politecnico di Torino, Italy. His current research activities focus on network protocols, traffic analysis, efficient packet processing and software-defined networks. Contact him at fulvio.risso@polito.it.

Riccardo Sisto, PhD in Computer Engineering, has been working at Politecnico di Torino, in the Control and Computer Engineering Department, first as a researcher, then as an associate professor and, since 2004, as a full professor of Computer Engineering. His main research interests are in the areas of formal methods and communication protocol engineering. Contact him at riccardo.sisto@polito.it.

Cecilia Corbi graduated in 1985 in Mathematics. With more than 25 years of experience in the telecommunications field, she has been working as Project Manager for cellular network operators in several areas of expertise in innovation and strategy. She is currently Application and Services Standards manager in Telecom Italia, and serving as Board Vice Chair of the Open Mobile Alliance. Contact her at ceciliamaria.corbi@telecomitalia.it.

Giorgio Ghinamo, graduated cum Laude in 1995 at Politecnico di Torino in Telecommunication Engineering. Researcher at Telecom Italia, his current interests refer to positioning technologies and related applications. He has been involved in navigation related standardization framework in OMA. Contact him at giorgio.ghinamo@telecomitalia.it.

Francesco Vadalà graduated cum Laude in 1996 from the Università di Messina in Electronic Engineering. Researcher at Telecom Italia, he has been working in different innovation/technical strategy and R&D areas. He analyses emerging technologies and coordinates standard activities in the areas of service layer. He is currently serving as Technical Plenary Chairman of the Open Mobile Alliance. Contact him at francesco.vadala@telecomitalia.it.

Pietro Leggero graduated in 2012 from Politecnico di Torino in Computer Engineering. Software architect and mobile application developer, specialist on Android, Java, RESTful WS, Geolocation, PAAS and NFC technologies, he develops and contributes to investigate on the new emergent mobile technologies. Head of Mobile development at SeekAnGoo. Contact him at p.leggero@mytaxi.net.

6 References

[1] Google, “The Google Directions API,” [Online]. Available: https://developers.google.com/maps/documentation/directions.

[2] Google, “Google Maps/Google Earth APIs Terms of Service,” [Online]. Available: https://developers.google.com/maps/terms.

[3] Open Mobile Alliance™, “OMA Dynamic Navigation (DynNav) v1.0,” 11 December 2012. [Online]. Available: http://technical.openmobilealliance.org/Technical/release_program/dynNav_v1_0.aspx.

[4] G. Ghinamo, F. Vadalà, C. Corbi, P. Bettassa, F. Risso and R. Sisto, “Vehicle Navigation Service Based on Real-Time Traffic Information,” in *Ubiquitous Positioning, Indoor Navigation and Location-Based Service*, Helsinki, 2012.

[5] Open Mobile Alliance™, “OMA Guidelines for RESTful Network APIs V1.0,” 2012. [Online]. Available: http://www.openmobilealliance.org/Technical/release_program/GuidelinesREST.aspx.

[6] R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[7] ISO TPEG, “ISO/TC 204 - Intelligent transport systems,” [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=54706.

[8] IETF RFC 4776, *Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information.*

[9] W3C, “Points of Interest Core,” [Online]. Available: http://www.w3.org/TR/2011/WD-poi-core-20110512/.

[10] Telecom Italia and Politecnico di Torino, “Vehicle Navigation Service Based on Real-Time Traffic Information,” [Online]. Available: http://openmobilealliance.org/wp-content/uploads/2012/12/Vehicle-Navigation-Service-Based-on-Real-Time-Traffic-Information.pdf.

[11] Telecom Italia and Politecnico di Torino, “Dynamic Navigation,” [Online]. Available:

http://www.openmobilealliance.org/comms/videos/telecom_italia_video_4.html.