Dynamic Cloud-based Vehicle Apps
Information Logistics in Disaster Response

Oleg Gusikhin[1], Ayush Shah[1], Omar Makke[1], Alexander Smirnov[2] and Nikolay Shilov[2]
Research and Advanced Engineering, Ford Motor Company, 20300 Rotunda Drive, 48121, Dearborn, Michigan, U.S.A.
[2]SPIIRAS, 39, 14 Line, 199178, St.Petersburg, Russia

Keywords: Disaster Response, Dynamic Vehicle Applications, Disaster Ontology, Connected Vehicle.

Abstract: The efficient management of transportation networks during disruptions caused by manmade accidents or natural disasters is a major attribute of the Resilient Smart City Transportation. There have been extensive research and development towards intelligent automatic disaster response systems. The majority of the proposed systems provide information logistics to the response team. In general, motorists caught in the disaster area typically tend to "go with the flow" or operate in an unorganized manner that may hamper the emergency response efforts. Connected vehicle technology and interactive vehicle applications enable the possibility to provide personalized information to individual motorists. This paper proposes the concept of dynamic vehicle applications integrated with cloud-based intelligent disaster response command and control system to facilitate evacuation, personalized routing, volunteering, and information gathering. The intelligent back end extends the knowledge based disaster response system for professional responders to automatically generate the guidance for the individual participant. The proposed dynamic vehicle applications leverage open source SmartDeviceLink interface and Node.js.

1 INTRODUCTION

Information Logistics is one of the crucial aspects of an efficient and effective evacuation and disaster response efforts. Information logistics is a field of business information systems that are focused on the concepts, methods and tools to provide the right information at the right time, in the right quality, in the right format and at the right place to the right actors (Smirnov et al. 2005b, Sandkuhl, K., 2008).

In the past two decades there have been significant R&D activities to address centralized control and coordination of disaster response activities resulting in the development of next generation command and control systems (e.g., Chaudhury et al., 2012). The majority of such Command and Control (C&C) systems target the support of professional responders. There has not been as much progress in the development of the efficient information logistics infrastructure for general population, specifically information logistics to the motorists caught in the disaster area.

The emergence of connected vehicles technologies enables to integrate the individual motorists with the C&C to further improve the efficiency of the disaster response. Such information logistics allow the increase of the evacuation efficiency by taking into account the attributes of individual vehicles, provide methods to receive automatic reports about the conditions of the road and surrounding area in the given location, and even incorporate volunteers into certain disaster response efforts.

Connected vehicles technologies can personalize evacuation, e.g. four-wheel drive truck or SUV may take certain shortcuts during an evacuation that may be inaccessible to front wheel sedans and sports vehicles. The evacuation route may also be affected by the available fuel, electric vehicle battery charge, and the availability of refueling and recharging options on the possible evacuation routes

In some situations, supporting certain areas is constraint by the availability of resources and the ease of accessibility of the areas. However, in many cases, the affected people do not strictly need professional assistance in order to resolve

Gusikhin, O., Shah, A., Makke, O., Smirnov, A. and Shilov, N.
Dynamic Cloud-based Vehicle Apps.
DOI: 10.5220/0006815606260635
In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2018)*, pages 626-635
ISBN: 978-989-758-293-6

their challenges, e.g., transportation from the disaster area, starting a vehicle with battery failure, or pulling a vehicle out of a gutter. In these cases, the participation of volunteers can fill the gaps, assuming these volunteers have access to the proper information and coordinate properly. This reduces the demand for professional rescuers who can concentrate on more critical locations. Nevertheless, the volunteers are usually unorganized, and in many cases their efforts to help cause more problems. For example, "…what they do is just to create more traffic on the roads and they impede the flow of the critical goods getting to the area" (Gehr, 2017).

Therefore, providing efficient means to incorporate the disjointed volunteer resources into a centralized disaster response C&C center has the potential to transform these resources to valuable assets to complement the efforts by the professional responders (or at least prevent these volunteers from "getting in the way") and increase efficiency of the response effort (e.g., FEMA, 2017).

This paper tackles the challenge of bringing information logistics and automation in disaster response to motorists in order to help them organize in an effective and efficient manner. The concept of "Dynamic Cloud-based Vehicle Applications" is introduced in section 4, which is a new type of vehicle applications, and an example is provided to demonstrate how these applications may function in real scenarios.

2 DISASTER RESPONSE AUTOMATION

Usually, disaster response operations involve a large number of different heterogeneous teams (sometimes multinational), which have to collaborate in order to succeed. Such teams may include medical brigades, firefighters, rescuers, military personnel, commercial / governmental / non-commercial organizations, volunteers, etc. Besides, during such operations it may be necessary to use external sources to get the required information (e.g., medical databases, transport availability, weather forecasts). Their coordination requires intensive information exchange in order to achieve the necessary level of the situational awareness, create ad-hoc action plans, and continuously update relevant information (Smirnov et al., 2010).

In (Smirnov et al. 2005a) a conceptual framework and hybrid technology for operational decision support based on the concept of information logistics was developed. The main idea behind the conceptual framework was using an ontology-based context model for modelling the current situation and solving incoming problems. Ontology properties of this model make the context a sharable model that can be accessible by its components for the purposes of exchanging and integrating the right information and providing knowledge at the right place and time. The context components are related to various resources of information and knowledge. Context properties of the context model enable the decision support system to process and interpret the dynamic information flow at the right time.

In (Smirnov et al., 2010) an approach to organizing resources in a smart environment for disaster response was proposed, and is shown in figure 1. It is based on the above conceptual model and assumes Web-services forming an ad-hoc service network according to the context of the current situation. The Web-services are responsible for producing real-time picture of the disaster situation, receiving diverse information resources, and problem solving. These Web-services provide an ontology-based model of a disaster situation that embeds models for problems requiring solutions in this situation. Harmonization of Web-service descriptions and the ontology allows the Web-services to exchange information about their needs and possibilities in terms of the ontology vocabulary. The functions of the identifying problems to be solved and distribution of actions or tasks between acting agents can be implemented in a unit called "task manager".

The ontology-based problem model represented as context following the idea of information logistics provides the Web-services with awareness about the problems to be solved in the disaster situation and information needed for this. The response members' profiles allow the Web-services to take into account specific information about the members as well as their tacit and explicit preferences. As a result, the Web-services become capable to form a service network for a common purpose (Smirnov et al., 2010). If the service assumes the presence of a human behind it, it renders information through an adaptive interface provided by a "front end generator" in the right form. For example, the "front end generator" can generate information

using HTML5 which can be viewed by operators using web browsers.

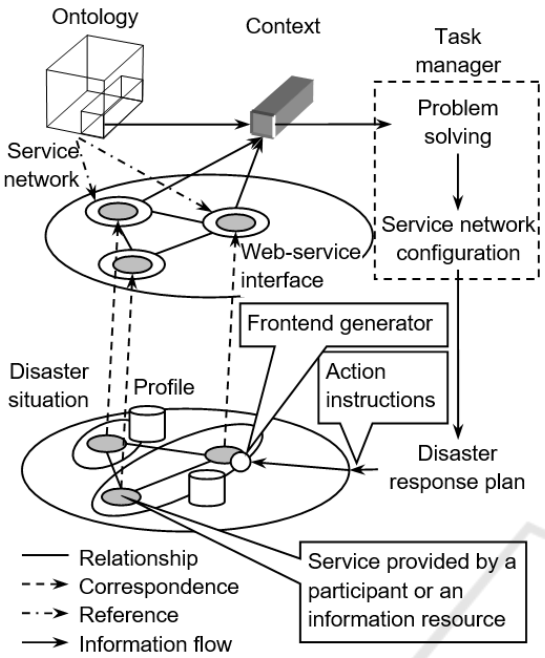

Figure 1: Ontology-based automated disaster response system.

Although the system in figure 1 was designed to address the information logistic needs for professional responders (Smirnov et al., 2010), this system can be extended to include the general population, specifically motorists.

3 CONNECTED VEHICLES SERVICES

Proliferation of connected vehicle technologies facilitates information support of the motorists caught in disaster area. In general, the term connected vehicle refers to the technologies that enable two-way communication between vehicle systems and outside world, including cloud, roadside infrastructure, and other vehicles. Today, there is a variety of different approaches to implement connected vehicle, such as OEM installed embedded modem, OBDII plug-in (Kolmanovsky et.al. 2011), integration with customer mobile phone (Yeung et. al. 2017), or DSRC Vehicle to Vehicle (V2V) or Vehicle to Infrastructure (V2I) technologies (Kenney 2011). In many cases the vehicle combines several of these technologies that allow improving robustness of telematics tasks through a multiprotocol connectivity manager (Zaborovski et. al. 2013).

There are a number of connected vehicle services to help the motorists in emergency or hazardous situations or environments. One way to provide the emergency assistance is through OEM installed embedded modem. For example, OnStar, ERA-GLONASS and E-Call utilize the vehicle's embedded modem to connect vehicle occupants to the dispatcher and transmit the vehicle status data. OnStar connects the vehicle driver with the OnStar advisor who can direct emergency services to vehicle location, and provide evacuation routes or other necessary assistance in case of accident, disaster or severe weather. ERA-GLONASS and E-Call are government regulated systems that connect the vehicle occupants to the local Public Safety Answering Point (PSAP) and transmit the critical data in case of accident. (Öörni et al. 2015).

Another way to provide assistance to the motorist is through mobile applications integrated with the vehicle's head unit. For example, KATWARN system provides smart phone application that can be integrated with Ford SYNC3 system using Applink. The service provides warnings and behavioral advices for the user's current location, as well as for seven freely selectable locations, and offers topic-related safety information quickly to the Ford SYNC GEN3 infotainment system (Katwarn, 2017; Ford 2017a). Another example of such app is HAAS Alert (HAAS Alert R2V, 2017). HAAS Alert alerts motorists when emergency responders are in the vicinity or en route, responding to a call. HAAS Alert has partnered with Waze to provide users with emergency incident locations and warnings when emergency services (Firefighters, Police, EMS) are on the scene to warn and/or re-route drivers to avoid collisions and delays. HAAS Alert has demonstrated Applink integration of their mobile application at SmartDeviceLink Hackathon 2017 (Ford, 2017b).

With the help of these applications, drivers can make decisions on how to avoid collisions, select alternative routes and reduce traffic delays. However, in order to take advantage of these applications the app needs to be already loaded into the driver phone.

These connected vehicle services provided the motorists with new methods to handle uncommon situations, and were a step closer to dynamic cloud-based vehicle applications.

4 DYNAMIC CLOUD-BASED VEHICLE APPLICATIONS

4.1 Motivation

The existing connected vehicle emergency services have certain limitations to support effective and efficient information logistics for motorists in the affected area. These services are either requiring preinstalled application or involve human operators that may increase response time, and are designed to solve specific problems. Moreover, the vast majority of mobile applications do not integrate with the vehicle's head unit, and therefore are not suitable for motorists to operate while driving, and therefore they lack in providing the right information in the right format and at the right place. The second section of this paper discussed ontology based systems for disaster recovery automation. The system operates in a command and control center and initiates tasks to concerned parties. The limitations of this system is extending it to motorists by providing the right information at the right time in the right place, whether they are volunteers or part of official rescue teams. This is due to the fact that the right time and the right place may depend on the vehicle's location, fuel range, and its field capabilities (towing, reverse electric flow, has jumper cables, etc.). The need to extend the ontology based system to include vehicle ontology, and to combine the system with connected services become evident. Dynamic cloud-based vehicle applications combine an extended ontology based system in the cloud with connected vehicle services to dynamically interact with motorists while considering their location, vehicle capabilities based on the current situation in their location.

4.2 Foundational Technologies

The specific example of implementing dynamic cloud based applications described in this paper utilizes open source SmartDeviceLink (SDL) and cloud Node.js as a reference platform.

SmartDeviceLink (SmartDeviceLink, 2017) is an open source project pioneered by Ford Motor Company that connects in-vehicle infotainment systems to smartphone applications allowing automakers the opportunity to provide customers with highly integrated connected experiences, and application developers with new ways of connecting with their customers (Yeung et. al. 2017). SmartDeviceLink is currently available on all Ford vehicles equipped with SYNC GEN3 infotainment or above, and will likely be integrated with the head units of other OEMs in the SDL consortium, such as Toyota., Suzuki, Mazda and Subaru. SmartDeviceLink enables mobile developers to integrate their applications with the vehicle's head unit. As of now SDL SDK's are available for Android and iOS mobile application developers. SmartDeviceLink uses Remote Procedure Calls (RPC) to interact with the vehicle system. This interaction includes showing information on the head unit, speaking through the audio system using text-to-speech embedded software, add buttons, list, and other interactive widgets to the head unit based on the content, and respond to the user's interactions. These RPCs are used to trigger a specific action defined by the application developer. The RPC is issued by the mobile application and the called procedures are executed by SDL. Notifications and responses to the RPCs are sent back from the vehicle's head unit to the mobile application. These RPCs use JSON format to exchange information between the vehicle and the mobile application. Unlike other In-Vehicle Infotainment systems (IVI), SDL RPCs are lightweight and language independent; any application developer can generate and trigger the defined RPCs in any language, even though only Android and iOS SDK are made readily available. Figure 2 shows a simplified architecture diagram of current implementation of SDL with Mobile devices. SmartDeviceLink's Mobile SDK generates all the RPCs on behalf of mobile Application.

These RPCs can be sent to the vehicle over Bluetooth, TCP/IP, or even through USB. RPCs are executed the IVI and creates a user interactive UI individually for each of the applications. SmartDeviceLink uses a template based approach where each OEM can implement a set of defined templates with respect to their driver distraction guidelines and policies. These templates define "what" widgets can be on the screen, and the OEMs define "how" these widgets are located on the screen. SmartDeviceLink SDK's can be extended to other platforms. For example, a C/C++ SDK can be developed to allow sensors to communicate with the vehicle and present an interactive interface on the head unit. For this paper, an SDK for Node.js has been developed to bring web applications to the vehicles. The popularity of programming in JavaScript has been

increasing recently, and more developers are becoming acquainted with Node.js (Stack Overflow 2017). SDL Node.js SDK issues the required RPCs in the cloud which triggers the respective reaction on the head unit directly, without requiring a mobile phone.

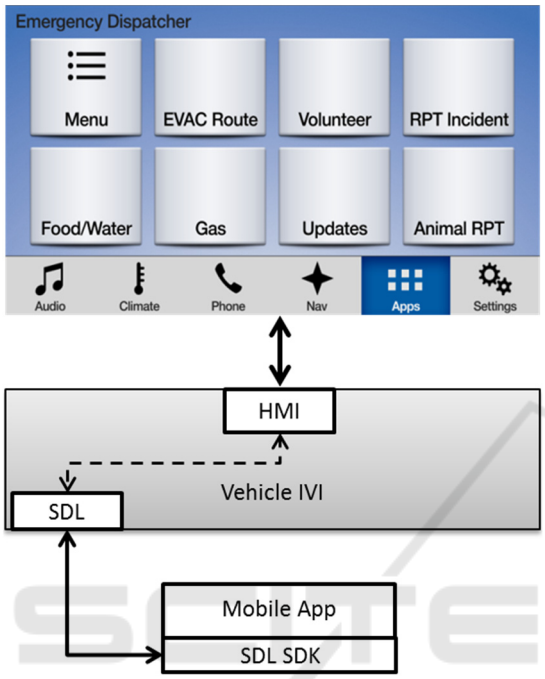

Figure 2: Current SDL Implementation with Mobile Applications.

SmartDeviceLink has security features which can be extended if needed to meet specific requirements. Its default security settings are based on policy tables which are securely downloaded and updated from the cloud, and which contain the access levels for each OEM approved application. During the process of approval, the OEM issues an application ID (App ID) to the developers. When SDL application starts communicating with the head-unit, the App ID is sent to the head unit. The head unit will then look in a local policy table, which resides in the head unit, to see which RPCs are allowed to be executed by the application, based on the App ID. The head unit checks for policy table updates on regular basis, and whenever a new App ID is detected. This allows the OEM to add or revoke permissions at any time. There are few methods to secure the App ID which are beyond the scope of this paper.

The local policy table residing in the head unit contains what the application can execute. SmartDeviceLink allows the customers to select which RPCs will actually execute, which allows users to have full control on what the application can actually do. For example, an application may be allowed to extract vehicle data, call 911 in case of emergency, stream music over Bluetooth, stream video for navigation applications, trigger alerts and popups, control radio, climate media or other modules. Figure 3 shows how SDL is typically integrated with a local policy table and the vehicle. This security mechanism applies to all SDL based applications, including the cloud based applications which will be discussed in the following section.

4.3 Cloud-based Vehicle Applications

The architecture of the cloud based vehicle applications is shown in figure 4. The Head Unit Service (HUS) is a service which runs in the vehicle and uses the in-vehicle connectivity manager to connect to an authentication server and Node.js server. The HUS can also be implemented in a mobile application and act as a relay between the cloud and SDL. The connectivity manager's core functionality is to provide connectivity to the vehicle using any available device or combination of devices, utilizing multipath TCP if necessary. (Chari et. Al. 2018). The connectivity manager may use a telematics unit and a mobile phone at the same time, or use DSRC V2I for example. The authentication server in figure 4 has a list of cloud applications and the required information to access of the Node.js server for a given user and VIN number, such as IP address and other authentication information. This information is used when the vehicle starts, to notify the cloud about its status and presence. The List of applications can be dynamically updated based on geographical location or time and date.

For example, when a driver approaches a drive-through to order food, an application for that restaurant appears on the application list in the IVI. The Node.js server is the server where multiple web SDL applications can be running. These applications are accessed and controlled as soon as the vehicle is started and HUS is initialized.

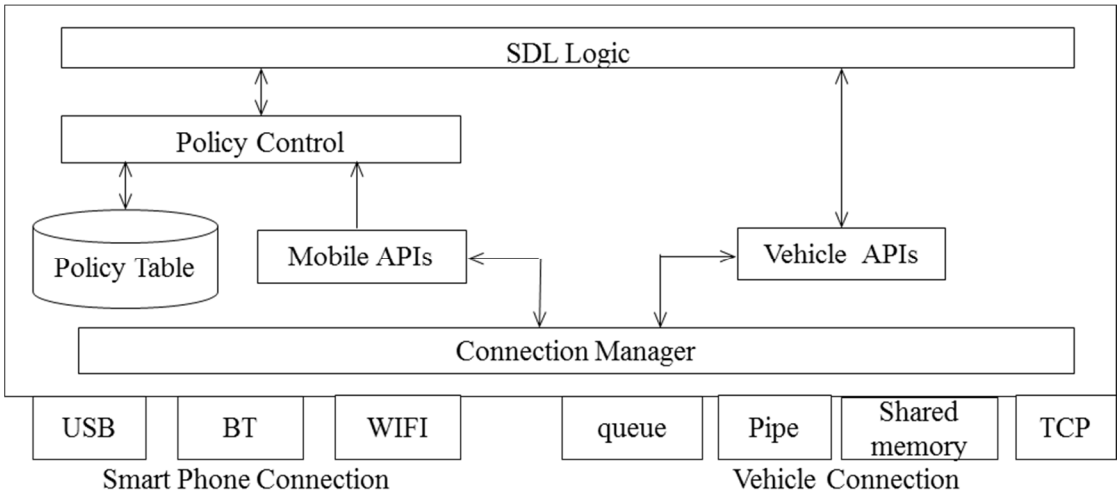

Figure 3: Typical SDL Integration Within a Head Unit.

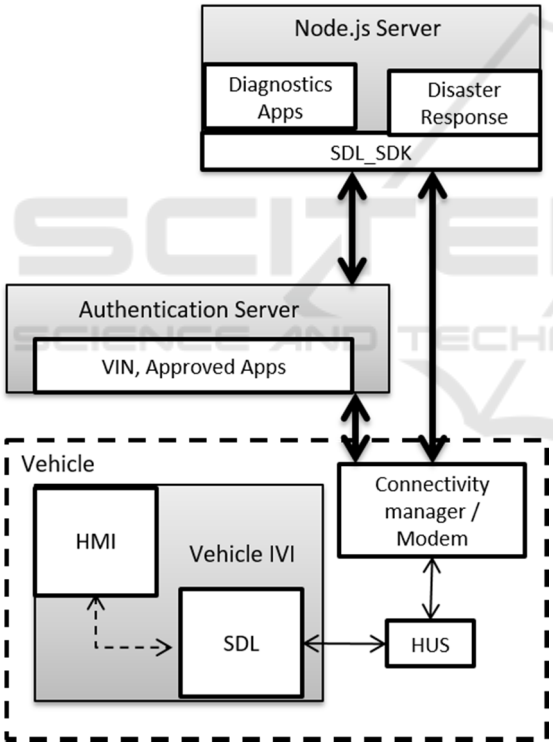

Figure 4: Web SDL Applications Architecture.

The system in figure 4 works as follows. A Node.js server is hosted in the cloud. The authentication server is populated with IP address/port, geo locations, and other necessary parameters to bring dynamic content. Whenever the vehicle is started, HUS creates a request with vehicle's VIN number and geo location and sends a request to authentication server. Authentication server will respond with list of application associated with VIN number and geo location. HUS will then use the list of application received from the authentication server and causes SDL to create a session for each application. Once session is successfully created, Node.js will start registration process for the applications. The registration process allows the applications to appear on the head unit, and from that point, the driver can start interacting with the application. SDL Node.js applications receive call-backs such as button pushes, or notifications about events, such as fuel level change, gear shift, etc. Developers can use these notifications to create an interactive cloud based SmartDeviceLink applications.

4.4 Dynamic Cloud-based Vehicle Applications

Dynamic Cloud Applications form a special class of web applications. The back end is ontology based context aware intelligent system. At its core is a Task Manager as seen in figure 5. It receives user interactions and real-time data as input, and using an ontology description of both the input data and the set of problems to be solved (the objectives), produces an optimized output. The optimized output is either a new Task description of a task to be solved, or a modification to a currently existing Task description. The Task description feeds into a front end generator which generates or modifies existing Node.js applications or the data associated with these applications. These applications are the Dynamic Applications.

A notable difference between this approach and the classical approach of web applications is that there is no front end code such as JavaScript to be downloaded onto the vehicle. The backend uses RPCs and the application on the user's device is rendered just-in-time, which makes the concept of dynamic applications possible. The benefits of using templates become evident at this point. The UI design elements, such as colors, font size, button locations, list menus, etc. is known beforehand (the selected template). This ensures that the requirements that the government regulations and recommendations and the company's driver distraction requirements are met. The use of templates also simplifies the process of dynamically generating an application because the UI layout is predefined.

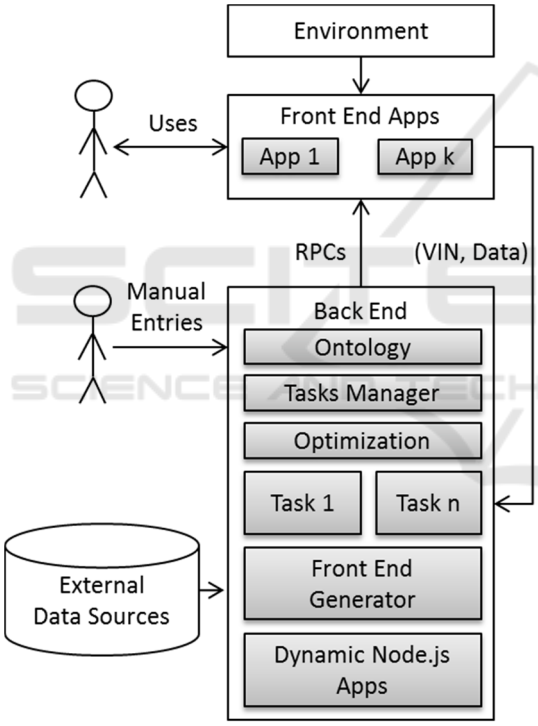

Figure 5: System structure.

Different geographical areas may have different objectives. As vehicles pass through an affected geographical area, the drivers are prompted through the Dynamic Node.js Applications to opt in to use the application to support in the disaster recovery efforts and/or to utilize the application's functions. Upon acceptance, the application displays several options to the drivers which are generated by the Front End Generator. An example is shown in figure 6. Drivers may use the application for their own benefit, such as to check gas prices, water and food availability, and in return, they provide valuable vehicle data for that period of time.

The drivers can also use the application to report incidents which are analyzed by the knowledge based system which, in turn, interacts with the Task Manager in the cloud and add to the objectives. The drivers may also select the option to volunteer, and then a menu appears, showing the tasks which are optimized for that vehicle. This is shown in figure 7.

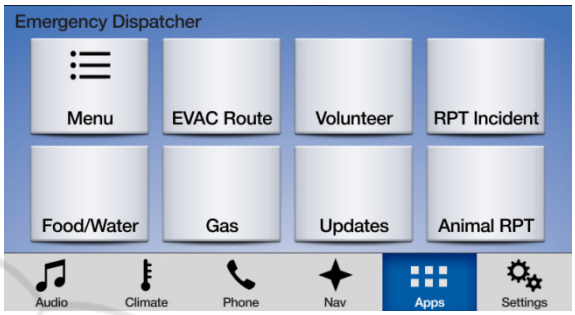

Figure 6: Dynamically Generated Main Menu.

If the driver chooses to jump start a vehicle, further information appears about the task. This is shown in figure 8. The driver may choose to navigate to that location, view next task, or go back to the menu. All this HMI logic is generated by the Front End Generator, and is communicated to the vehicle using RPCs.

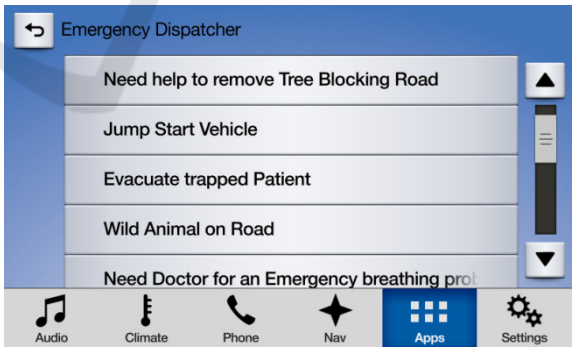

Figure 7: Dynamically Generated Tasks List

The benefit of using SDL becomes evident at this point. The application appears to be integrated with the vehicle, and the data is presented in a controlled fashion. The driver may choose to navigate to the location. In this case, the address, via SDL, is pushed to the embedded navigation system if available. The driver may also call a

phone number provided through the backend using the vehicle's hands free system.

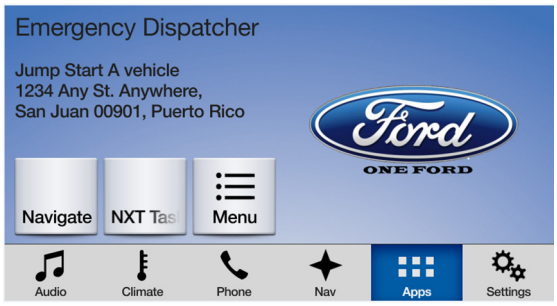

Figure 8: An Example of Dynamically Generated Task Information.

The Task Manager in figure 5 is the major component of the back end system. It is the component which uses the real-time input from the connected vehicles and users who are involved in the disaster recovery efforts. The data received contains sensory information such as rain sensor data, fuel level, temperatures, dust, pressure, camera, etc. and user information such as a list of available equipment, VIN number which maps to vehicle's capabilities (towing, all-wheel drive, …) volunteering time window etc. This information is received by the Dynamic Application and is forwarded to the Task Manager for use.

The Task Manager can also connect to external data sources such as weather prediction for example, or retrieve information about functioning gas stations, clinics, etc. If necessary, the Task Manager can update any data table associated with Tasks, which then immediately feeds into the Front End Generator and update the Dynamic Node.js Application's behavior and data in real-time. For example, a user may be scrolling through available tasks using the head unit, and then new items can be added to the scroll menu. It produces a Task description, which is an implementation independent description of tables and information required for a UI to be rendered. This is usually done as a tree data structure. The Front End Generator uses this data structure to generate new or modify existing code and data in the Dynamic Node.js Application. In summary, the Task Manager is the component which, based on ontology and available information, oversees the behavior of the entire system, independent from vehicle's implementation details.

The Dynamic Node.js Application which is generated by the front end generator is what appears to be as context aware discoverable service. The Dynamic Node.js Application contains all the possible tasks which the vehicle of a known VIN number with a given user can achieve. An F150 truck has more capabilities than a Ford Focus, and if towing capability is required, the task will not appear in the application within the Ford Focus. If there are two F150's, the fuel level and GPS location can be used to find out which truck is most suited for what task at known locations.

5 ILLUSTRATIVE EXAMPLE

In this section, an illustrative example of how this system is utilized in the case of a stranded vehicle due to a dead battery is described. The system receives information about the stranded automatic transmission vehicle with two people due to dead battery through its web interface. This information, shown in figure 9, is processed based on the system's internal ontology, and it represents a fragment of the ontology describing the "Accident" concept). Then, the system identifies the appropriate available resources based on the situation description: for example, requirements of a jumper cable, the battery specifications, location, and instruction on how to start it and space for two extra passengers in case the battery cannot be jump started.

The services involved and their interactions are shown in figure 10. It has two sub scenarios: volunteer registration and disaster response. The volunteer registration scenario takes place when the vehicle enters the area affected by the disaster. First, the gateway ping will return the reference to the emergency app. The vehicle head unit will access the application and the alert will be generated that new emergency assistance application is available. Then the driver opts in and the head unit will dynamically load the task description in the application. The example of the application is shown previously in figure 8. The specific implementation and UI may be different for different municipalities and types of the disaster.

Similarly, using this application, it is possible to get road assistance, best personalized evacuation route; find the best place to get gas based on your current gas level; find the info and route where to get food and water; report accident; report animal danger; or volunteer to provide help to other people.

Figure 9: A fragment of disaster response ontology related to the "Accident" concept.

6 CONCLUSIONS

In this paper, connected vehicle technologies such as dynamic cloud-based vehicle applications are shown to enable the possibility to cohesively integrate disaster response efforts of individual motorists with existing rescue operations according to their individual capabilities. A novel approach to dynamically generate information which conforms to information logistics philosophy is discussed. This system is ontology based context aware system which maps problems to achievable objectives and assigns them to specific motorists based on their profile and feedback. The motorists can interact with these objectives using dynamically generated applications which are made possible by leveraging the capabilities and synergies of Node.js and SmartDeviceLink. The presented approach extends the information logistics of cloud based system to vehicle environment, which allows a better coordination between first responders' efforts and all other motorists in the affected area. Enhancements to this platform can be made in the future by mitigating the effects connectivity issues. In theory, it is possible to mobilize specialized vehicles which have local cloud infrastructure, which introduces the concept of cloud-on-vehicle. Using V2V such as DSRC, or V2I, the connectivity range can be extended by having the vehicles out of connected regions to communicate with a moving cloud-on-vehicle system. The vehicle would move to areas with connectivity on daily basis to download new local updates for the area, and then moves to an affected connectionless area. A further investigation about the feasibility of this approach is required, and to see how connectivity plays in the geographical topology of the affected area to make this system possible.

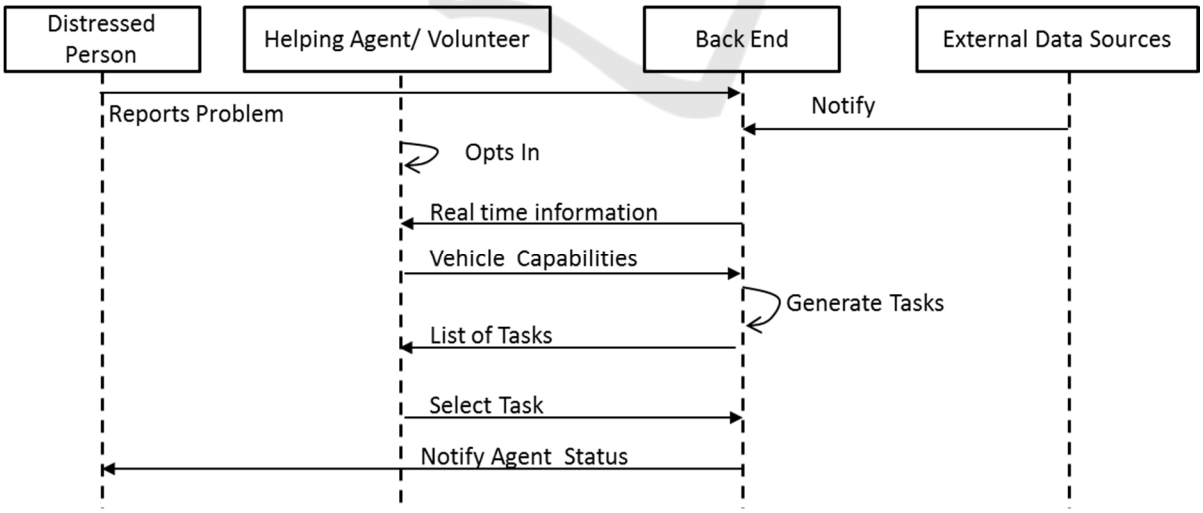

Figure 10: Diagram showing the interactions between different components.

REFERENCES

Chari, V., Tonshal, B., Kondoju, P., and Gusikhin, O., 2018, Vehicle Telematics Platform using Multipath TCP, SAE Technical Paper 2018-01-0753, 2018.

Chaudhury, K., Nibedita, A., & Mishra, P. K. 2012. Command And Control in Disaster Management. IJCSI International Journal of Computer Science Issues, 9(4), 256-259.

FEMA, 2017. How to Volunteer for Hurricane Irma Disaster Relief. Accessed on 18.12.2017 at https://www.fema.gov/news-release/2017/09/09/-how-volunteer-hurricane-irma-disaster-relief.

Ford, 2017a. *Ford holt vernetztes Warnsystem KATWARN ins Auto*. Accessed 14.12.2017 at: https://media.ford.com/content/fordmedia/feu/de/de/news/2017/02/28/ford-holt-vernetztes-warnsystem-katwarn-ins-auto.html.

Ford, 2017b. Inaugural SDL Hackathon a Success! Accessed 14.12.2017 at: https://developer.ford.com/-pages/inaugural-sdl-hackathon-a-success.

Gehr, D., 2017. Hurricane Harvey volunteer mistakes cause more harm than good. Accessed on 18.12.2017 at http://www.iowastatedaily.com/news/-nation/article_e4ec7a46-8d01-11e7-bcef-3becba27581c.html.

HAAS Alert R2V, 2017. Accessed 14.12.2017 at: https://www.haasalert.com/.

Katwarn, 2017. The Warning and Information System for the Public. Accessed 14.12.17 at https://www.katwarn.de/downloads/en/-KATWARN.pdf.

Kenney, J.B., 2011, Dedicated Short-Range Communications (DSRC) Standards in the United States. Proceedings of the IEEE, vol. 99, no. 7, pp. 1162-1182.

Kolmanovsky, I., McDonough, K., Gusikhin, O., 2011, Estimation of fuel flow for telematics-enabled adaptive fuel and time efficient vehicle routing, 11th International Conference on ITS Telecommunications (ITST 2011), pp. 139 - 144

Öörni, R., Meilikhov, E., Korhonen, T. O. 2015. Interoperability of eCall and ERA-GLONASS in-vehicle emergency call systems, IET Intelligent Transport Systems, 2015, Vol. 9, Issue 6, pp. 582–590.

Sandkuhl, K., 2008. Information Logistics in Networked Organizations: Selected Concepts and Applications. In: Filipe J., Cordeiro J., Cardoso J. (eds) Enterprise Information Systems. ICEIS 2007. Lecture Notes in Business Information Processing, vol 12. Springer, Berlin, Heidelberg, 43-54.

SmartDeviceLink, 2017. *SmartDeviceLink*. Accessed 14.12.2017 at: https://www.smartdevicelink.com/.

Smirnov, A., Levashova, T., Kashevnik, A., Krizhanovsky, A., Shilov, N. 2010. Self-Organisation of Smart Environment Resources for Disaster Management and Relief. In: Crisis Management (Patrick Alvintzi and Hannes Eder, eds.), Nova Publishers, 371 pp., ISBN: 978-1-60876-570-6, 2010, pp. 157-196.

Smirnov, A., Pashkin, M., Chilov, N., Levashova, T. 2005a. Constraint-Driven Methodology for Context-Based Decision Support. J. of Decision Systems, Lavoisier, Vol. 14, No. 3, 2005, pp. 279 – 301.

Smirnov A., Pashkin M., Levashova T., Chilov N., 2005b. Fusion-Based Knowledge Logistics for Intelligent Decision Support in Network-Centric Environment. In: George J Klir (Ed.) *International Journal of General Systems*. Taylor & Francis, 34(6), pp. 673-690.

Stack Overflow, 2017. Stack Overflow Developer Survey 2017. [ONLINE] Available at: https://insights.stackoverflow.com/survey/2017. [Accessed 15 December 2017].

Yeung, J., Makke, O., MacNeille, P., and Gusikhin, O. 2017. SmartDeviceLink as an Open Innovation Platform for Connected Car Features and Mobility Applications, SAE Int. J. Passeng. Cars – Electron. Electr. Syst. 10(1):231-239, 2017

Zaborovski, V., Chuvatov, M., Gusikhin, O., Makkiya, A., Hatton, D., 2013, Heterogeneous multiprotocol vehicle controls systems in cloud computing environment. Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2013) Vol. 1, pp. 555 – 561.