Cloudthink: a scalable secure platform for mirroring transportation systems in the cloud

Erik Wilhelm[1], Joshua Siegel[2], Simon Mayer[3], Leyna Sadamori[4], Sohan Dsouza[5], Chi-Kin Chau[6], Sanjay Sarma[7]

[1]*Singapore University of Technology and Design, Singapore*
[2, 7]*Massachusetts Institute of Technology, United States*
[3, 4]*ETH Zürich, Switzerland*
[5, 6]*Masdar Institute of Science and Technology, United Arab Emirates*

Abstract. *We present a novel approach to developing a vehicle communication platform consisting of a low-cost, open-source hardware for moving vehicle data to a secure server, a Web Application Programming Interface (API) for the provision of third-party services, and an intuitive user dashboard for access control and service distribution. The CloudThink infrastructure promotes the commoditization of vehicle telematics data by facilitating easier, flexible, and more secure access. It enables drivers to confidently share their vehicle information across multiple applications to im-prove the transportation experience for all stakeholders, as well as to potentially monetize their data. The foundations for an application ecosystem have been developed which, taken together with the fair value for driving data and low barriers to entry, will drive adoption of CloudThink as the standard method for projecting physical vehicles into the cloud. The application space initially consists of a few fundamental and important applications (vehicle tethering and remote diagnostics, road-safety monitoring, and fuel economy analysis) but as CloudThink begins to gain widespread adoption, the multiplexing of applications on the same data structure and set will accelerate its adoption.*

Keywords: cloud theory; eSafety; information system; intelligent transport system; data security.

Introduction

Physical objects are increasingly being networked and projected as virtualized objects into the 'cloud', an amorphous term that describes how data manipulation, platform development, and computing infrastructure construction may be provided as services. Cisco estimates that as of 2008 there were more networked objects than people on the planet, and predicts that in 2020 there will be 50 billion interconnected nodes (Evans 2011). Despite the proliferation of networked objects, a true 'Internet of Things' has not been realized due to the lack of effective standards and lack of a compelling business case for networking the world. While Internet Protocol standards have been developed, modified, and extended to accommodate the first few billion connections, these standards are written for virtual, rather than physical objects, and hence there exist many untapped opportunities to leverage big data to improve quality of life. This is also the case in the automotive domain. While many applications with networked vehicles already exist under the umbrella term 'telematics' (Gerardo, Lee 2009), they are often implemented as insular solutions, prohibiting synergies and economies of scale by restricting access and offering limited inflexible parameters to developers. For example, GM's OnStar, the most widely adopted cloud telematics platform, was only opened to approved developers. There are currently only two applications available in the gallery, and the barriers to become a new entrant and access vehicle data are severe. Ford's OpenXC platform is closer to a truly open method for moving vehicle data to the cloud, but is essentially limited to Ford's vehicles for turnkey operation, and does not leverage existing vehicle to cloud connections such as Ford's Sync. This platform does have the benefit that multiple hardware vendors offer connected devices. Finally, various start-up companies have begun to be active in the space, for example, Automatic offers a Bluetooth-link vehicle telematics system and Moj.io offers a GSM-based telematics platform, both with various developer interfaces and open Application Programming

Interfaces (APIs). Existing platforms fail to adequately address the fundamental tension between flexibility to host many applications, and the need for security and privacy in data transmission.

Against this backdrop, this paper outlines the development of a transportation-specific platform called CloudThink. CloudThink connects the world's vehicles and utilizes the additional information provided by networked automotive sensors in order to reduce energy use, cost, and environmental impact, and increase driver safety. Not only does CloudThink promise to enable many third-party application builders by lowering their barriers to entry for accessing vehicle data, but it will also give research groups access to rich data sets that will allow them to address important scientific questions. Aggregated sensing data from representative vehicle fleets is valuable for the understanding of human mobility patterns (González *et al.* 2008), the augmentation of road models (Rogers *et al.* 1999), traffic monitoring (Shi, Liu 2010), and the optimization of electric vehicle designs (Smith *et al.* 2011).With respect to travel and driving behaviour, vehicle data furthermore enables the setting of economic incentives – such as adaptive insurance rates or road pricing (Litman 1997), and the improvement of individual driving style through feedback (Toledo, Lotan 2006). There is a strong latent demand for this type of data evidenced by the vast number of connected car services available and on the horizon, and it is expected that the CloudThink API will need to handle many requests from project partners and act as a lightning rod for multiple spin-offs and research projects.

This technology platform is being designed to spark the development of standards for moving data from cars (and other objects) into the cloud. The evolution of a series of standards will be driven in part by user adoption of the CloudThink service, and hence an application ecosystem that provides compelling convenience and monetary motivation to join was considered important in the design of the system. CloudThink is an unbiased data broker that ensures customers' data is securely transferred to application clients who process the data to produce benefits for drivers. In doing so, it highly emphasizes user privacy, an important yet often neglected issue particularly in telematics (Duri *et al.* 2004, 2002; Iqbal, Lim 2010; Musicant *et al.* 2010). The strategy guiding the research and development of CloudThink is modelled after the highly successful approach taken by S. Sarma (2004) to standardize RFID technology in founding the Auto-ID labs and its corresponding industry organization EPCglobal which provided a much-needed link between research and industrial commercialization.

1. System Overview

To succeed in becoming a de-facto telematics standard, CloudThink depends to a large part on a swift, widespread adoption from drivers, vehicle manufacturers, research teams and third-party developers. To achieve the necessary critical mass, it is important to make the CloudThink service attractive to these key stakeholders by addressing platform needs from different perspectives. Researchers, third-party developers and vehicle manufacturers have in most cases already developed their own telematics solutions, albeit in a closed and domain/application-specific form (i.e. specifically for finding parking, predicting electric vehicle range, ensuring vehicle security, etc.). The CloudThink API and hardware were designed to offer distinct advantages in cost (using commodity components, design-for-manufacture, and design-to-value to keep hardware costs down and highly scalable), simplicity (plug and play interaction), security (state of the art HTTPS RESTful API), and scalability as well as with ease of porting existing solutions in mind. To appeal to drivers, joining the CloudThink service should have a marginal cost, clear benefit, and allow users maximum control of their personal data which is to be stored using state-of-the-art security measures.

CloudThink was engineered to maintain a careful balance of flexibility for new applications to request new data streams versus driver privacy and data security. Three key aspects of CloudThink differentiate it from existing telematics systems:

1. *Cost-effective, open-source hardware:* a new vehicle telematics hardware system has been developed, called the CARduino, which can efficiently and reliably transmit vehicle data in a 'plug-and-forget' way. The details of the hardware construction are open-sourced to allow third parties and researchers to build advanced functionality as required. Unlike existing systems, this open hardware has provisions for accessing broader vehicle data at the physical layer (Layer 2 CAN) which go well beyond what conventional on-board diagnostics platforms allow, and enables direct-to-cloud GPRS data streaming. Additionally, a smartphone-based software system has been developed to allow vehicle data to be pushed to the database using off-the-shelf OBD-Bluetooth adapters (Tahat *et al.* 2012). This software currently contains core functionality, including cloud-based data retrieval, a basic dashboard, charting and mapping, which can be adapted and extended for future developers' purposes. This hardware accesses a limited data set as compared to the CARduino, but allows users to interact with the CloudThink platform without dedicated data plans.
2. *Open Application Programming Interface (API):* the interface that application builders can use to access the vehicle data will also be open source to allow for a deeper understanding and scalable co-development. Individual user data, however, will be encrypted and securely stored. By standardizing the type and format of data accessible through the cloud-based Web API, and by creating a mechanism for applications to request new data be collected at the vehicle level as well as processed data be available in the cloud, appli-

cation builders are relieved from having to consider various hardware variants.

3. *Web-based, centralized user dashboard:* CloudThink gives the user full control over his or her data, however some data sharing may be a prerequisite to amortize the hardware cost in some scenarios. Through an intuitive dashboard, users can select services, administer payments and subscriptions, overview their stored vehicle data, and grant access rights to applications that handle it via the CloudThink platform. Transparency of which applications access a user's data is a major characteristic of the CloudThink ecosystem.

A low-cost modular data collection system called the CARduino was developed to efficiently and securely transmit vehicle data to the CloudThink Data Server so that virtual vehicles can be projected into the cloud for registered users, which will be described in more detail in the following section. This data collection system moves information first to the CloudThink Data Server, where it can be accessed and processed by authorized third parties through the CloudThink Gateway Server. The results can then be fed back to the user interface (laptop, smartphone etc.) constituting a third-party service. The advantage of this approach is that the data becomes multi-use between various services and the reliability of data collection increases substantially over systems where the data is passed directly to the user interface (via Bluetooth etc.), as user reliance and system complexity are minimized.

The system architecture shown in Fig. 1 has been created using REST Web APIs to maintain compatibility with widely deployed tools used by the Web community. Its main constituents are the CARduino (described above), the Data Server, which writes data that is uploaded from individual cars to a central database, and the Gateway Server, which provides access to the obtained data for third parties via a RESTful API (cf. Guinard *et al.* 2011).

1.1. Open Telematics Hardware

The open-source 'CARduino' interface shown in Fig. 2 is designed to be low-cost and highly flexible to accommodate the vehicle CAN data requirements of a very broad range of applications. The CARduino utilizes the

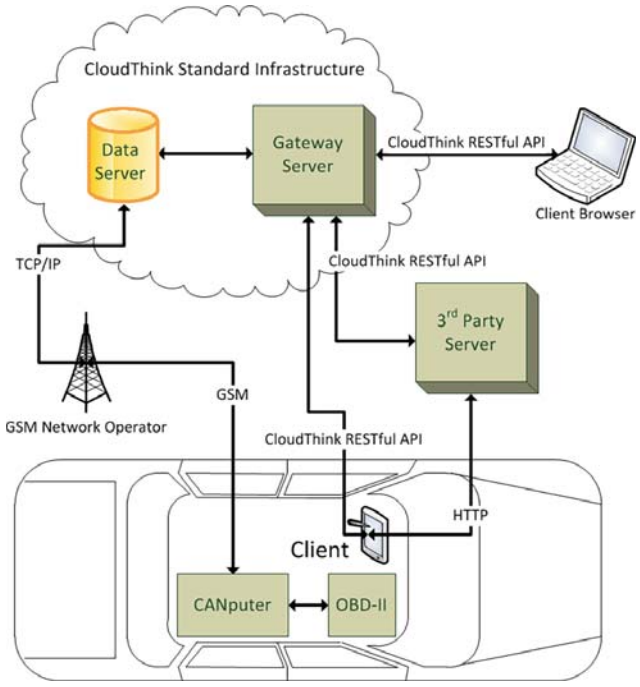

Fig. 1. CloudThink system architecture showing how the CARduino and the CloudThink Standard Infrastructure interact with 3rd party servers

standard OBD-II data found in all modern vehicle diagnostics systems and adds flash memory for data buffering and long-term storage, a GSM connection for near real-time communication, a high accuracy GPS receiver for accurate positioning data, and an accelerometer/gyroscope to provide anchoring data for motion-intensive studies. These data are transmitted through a GSM connection for quickly updating time-sensitive parameters. The on-board buffer is transmitted and cleared at the end of a trip to provide richer, higher frequency data in interstitial gaps left due to bandwidth constraints or poor network connectivity.

The CARduino is designed to be user-friendly, reliable, protective, secure, and low-cost to facilitate mass adoption. The hardware is user friendly, with no external buttons and only simple status indicator lights to indicate success or faults and is designed to be 'plug-and-forget'. An on-board buffer ensures reliable reporting of data, providing a complete dataset despite cellular intermittency. A Power-On Self Test (POST) routine self-

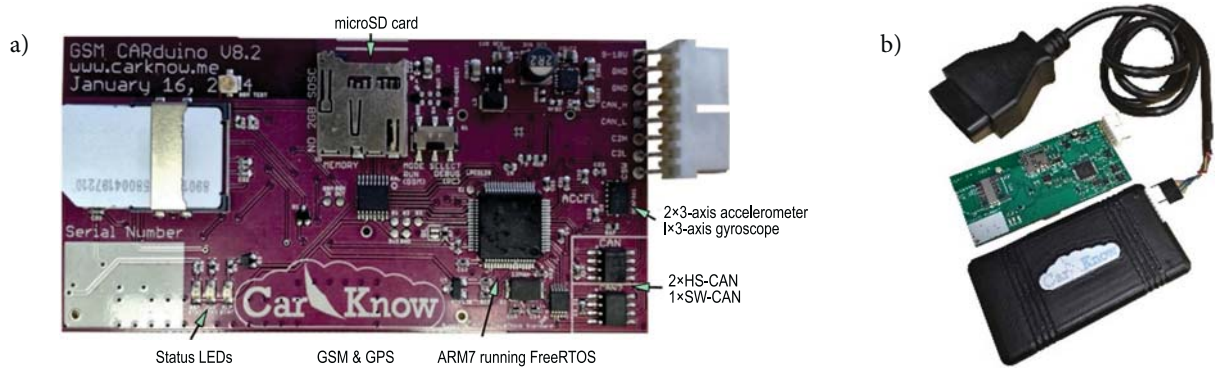

Fig. 2. CloudThink's v8.2 low-cost, open-source CAR-duino for moving vehicle CAN/OBD-II data into the cloud (a) – each device has approximately these dimensions: 9.3×4.4×1.5 cm; connection and case of the CARduino (b)

diagnoses faults, while a secondary flash-memory boot loader provides a redundant upgrade path in the event of a failed remote update event. User data is protected through the use of SSL encryption, and the hardware itself is resistant to spoofing when the external sensor payload is used to provide situational data to be compared against a reference.

Part of the reason for the CARduino's low-cost is a focus on value driven design, where each feature is designed-in only if it is deemed to be worth the price differential. Thus, while other solutions – such as OnStar or Coyote Systems – design hardware and software functionality for specific proprietary networks, or devote significant engineering resources towards supporting single applications, the CloudThink hardware focuses on transferring only the most critical data securely and reliably to facilitate a highly flexible and extensible software platform.

The CARduino's core functionality is to serve as a vehicle-to-cloud interface, bringing engine and transmission data from the diagnostic port directly to the secure database described in the previous section. While diagnostic scan tools exist today using Wi-Fi and Bluetooth, these networked devices embody a flaw inherent to their short-range communication. For mission-critical metrics, short-range devices present challenges as they rely on a rebroadcasting device. The CARduino avoids this pitfall by reporting directly to the CloudThink back-end using an integrated cellular chipset.

Using CloudThink CARduino hardware, the vehicle becomes a self-reporting sensor, requiring no user intervention after initial configuration. Further, the hardware is capable of reading non-OBD CAN messages from the vehicle as well as receiving actuation commands from the CloudThink server and transmitting the appropriate command to specific nodes on the vehicle network. The complexities associated with these non-legislated parameters have historically prevented application builders from taking advantage of the information present on the in-vehicle networks. While competing telematics systems ignore proprietary data acquisition or actuation, or focus on a single make of vehicle, the CloudThink hardware offers the ability to connect to secondary vehicle networks using a generalized, community-driven set of commands and parsing profiles curated similarly to a Wiki. By providing a hardware interface capable of supporting communication with these secondary networks, the CloudThink hardware supports extension beyond conventional diagnostic data. This level of extensibility allows new platforms to be added to the API without requiring changes to the embedded software functionality. Further, by creating direct access to secondary networks, a situation in which OEM gateway devices become overloaded and inadvertently deny service to networked devices may be more easily avoided. A command whitelist approach, as well as a firewall blocking access to non-approved servers, addresses the attack-scenario where the CloudThink platform is used to directly access vehicle networks.

By making the access to data transparent, it becomes easier to build applications and to focus on innovation rather than forcing developers to struggle with complicated standards and worrying about secure data transfer. CloudThink abstracts data acquisition to simplify capture and analytics, allowing developers to focus on their core competencies. It is worth emphasizing that no third-party server will be authorized to collect or transmit data using the CARduino, minimizing potential access points to sensitive data and safety-critical subsystems.

The CARduino was designed using the CAN standards specified in SAE (2010) Standard Collection HS3000 (J1962, J1978, J1979, J2284) and related ISO standards (ISO 15765-2:2011, ISO 15765-4:2011 and ISO 15031-5:2015), including 11- and 29- bit identifiers at 250 and 500 kbps (the CARduino does not support legacy diagnostic networks as the added value was deemed minimal). The hardware can access real-time and freeze frame PIDs, as well as diagnostic trouble codes at a rate of six unique samples per second using an ARM7 processor to do local data processing. These processors are among the most widely used ARM cores, keeping costs low and offering opportunities for incorporating additional peripherals without an architecture overhaul. In addition to accessing these vehicle data, the CARduino features a 48 channel GPS receiver and a three-axis accelerometer, as well as a microSD card for remote software upgrades and data buffering and storage. The CARduino plugs into a standard J1962 diagnostic port connector and is 24V tolerant, reverse polarity protected and fused to improve hardware robustness and support heavy duty vehicles with passive adapter cables.

As with the API, the CARduino embedded hardware and software source documents will be released to allow developers and end users to contribute to the code base. The CARduino's software may be compiled using free tools, and uploaded to the hardware for non-commercial purposes using freeware.

The CARduino aims to create a means of providing universal access to vehicle data. This requires more than simplifying access to 'conventional' diagnostic data, such as OBD PIDs and DTCs, it rather brings into focus access to data that conventional scanners are unable to read such as manufacturer specific diagnostic and configuration data. The CARduino can also facilitate access to physical layer data on the CAN bus, and is easily extensible as manufacturers elect to share additional information about their proprietary networks with the CloudThink platform.

Beyond serving as a scan-tool with remote data transfer, the device incorporates several innovative features. Diagnostic scan tools and modern data loggers are not intended to be left in cars. These tools must be unplugged to avoid draining a car's battery after use, whereas the CARduino intelligently determines when the driver is done using the vehicle and turns off key features to save power. The device similarly detects when the driver returns to the vehicle and resumes operation, lowering consumer interaction requirements and consequently barriers to entry.

The open hardware schematics and firmware are available from Internet: *http://www.carknow.me*.

1.2. Smartphone Open Data Channel

As a second channel for moving data from vehicles to the cloud using the CloudThink infrastructure, a smartphone data channel has been created. The smartphone-based software system supports the developer community in creating applications which make use of existing smartphone computing power to access, process, and serve vehicle data on-board and in the cloud. This section of the paper describes the smartphone application developed and open-sourced to allow users to read vehicle and smartphone sensor data, store and perform computations on the data local, and communicate with the remote CloudThink Data Server for data upload and download. As an open-source project, it is designed to allow developers to modify basic functionality or create additional functionality easily.

The applications run on a smartphone paired with a compact ELM327 adapter that is mounted to the vehicle's on-board diagnostics data port. The adapter contains an ELM327 microcontroller, or a widely- and relatively cheaply-available clone thereof, ELM327 being the predominant command protocol for open access to the OBD data of a vehicle.

Currently, there is a facility for the use of a Bluetooth connection – one of the most common OBD port adapters on the market – which facilitates serial communication between the smartphone and the adapter, allowing the former to query the latter for data from the vehicle's internal computer network, using a standard set of command codes.

However, other modes of communication can also be supported e.g. over Wi-Fi. Other protocols can also be used with appropriate customization of the communication module in the application source code, using specialized adapters, such as OpenXC or direct CAN access (Fig. 4).

When the respective listeners in the application receive data for the position/acceleration and OBD-II data from the smartphone sensors and an OBD dongle plugged into the vehicle, they update an internal data object in a running register and upload cycle, which, at the user-specified frequency, converts the data to the CloudThink upload format, and transmits it via a TCP connection to the CloudThink back end. The data upload cycle operates such that if it is collecting data when the user has not enabled upload, or has enabled upload by Wi-Fi only when the phone is not connected to a Wi-Fi provider, it will store the data on schedule, in memory for the short term, and in internal private storage files in the long term, until such time applicable connectivity is restored, or upload is enabled. The data is also backed up in periodically-rolled files that are stored on the phone's external storage. As per the CloudThink standard upload format, uploaded data parameters are individually time-stamped, independent of the timestamp of the upload itself.

The application's user interface displays useful information to the user, including primary data such as velocity, Revolutions Per Minute (RPM), fuel level, coolant temperature, and secondary data like the running and

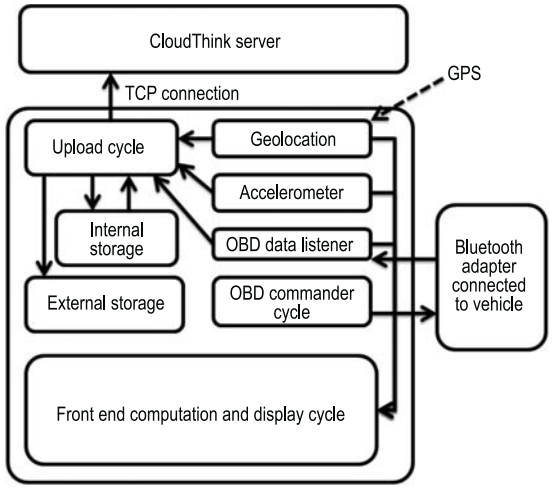

Fig. 3. Overview of the CloudThink smartphone-based software system architecture (the same data flows described in the API section have been recreated in this SmartPhone application)

a) b)

Fig. 4. User interface of the mobile application on a Galaxy S2 smartphone: a – dashboard screen; b – configuration options

average fuel consumption estimates, computed in real time using available engine data. Libraries for drawing charts and displaying maps have also been included in the application, permitting developers to adapt them to their needs while using the accumulated on-board data from the vehicle and/or the online data from the CloudThink API to present advanced functionality to users, not limited to long-term data analysis and even social applications.

1.3. Back End: Data Server and Gateway Server APIs

The following descriptions make exclusive reference to the CARduino data pathway for simplicity, although they could have as easily referred to the Bluetooth smartphone data channel described in the previous section. The CARduino transmits encrypted vehicle data to the Data Server via the cellular network, which stores it (again, encrypted) in an SQL-based database, that con-

tains a table for every car in the system. For the upload from the CARduino to the Data Server, we use a proprietary message format to minimize the volume of the transmitted data (and, thus, communication costs). A single message usually contains the recording timestamp of the CARduino, the type of the transmitted data (GPS/ACC for GPS/accelerometer readings, or the OBD PID of the data), the data itself, and a checksum. The Data Server unpacks and checks every message before storing its payload in the car's database table. In a typical application scenario, the vehicle's CAN bus may be sampled at 1 Hz for velocity, throttle, RPM, and mass airflow data. With accounting for various identification, time stamp, and integrity checking overheads, this represents roughly 12 bytes of data. If it is assumed that a vehicle travels for 2 hours per day, the US vehicle fleet would theoretically generate 8640 GB of data per day, and would require 9600 Mbps bandwidth. These figures certainly indicate that for widespread penetration of CloudThink open telematics systems both the 'volume' and 'velocity' criteria of big data cloud systems are met. Considering the large range of parameters available on the OBD-II bus, together with what is available at the deeper Layer 2 CAN bus, the 'variety' criteria is also satisfied. The CloudThink servers are provisioned to allow more resources for data streams from vehicles to be stored than to serve API queries. This is designed with the assumption that most applications will either query small volumes of real-time data, or large volumes of data in batches.

The Gateway Server arbitrates which data client applications, browsers, and third-party servers have access to, and provides a REST API that features encrypted connections (Secure Socket Layer/Transport Layer Security, SSL/TLS) and HTTP-based authentication. Data can be retrieved in a variety of formats such as JavaScript Object Notation (JSON) and Extensible Markup Language (XML), and can be navigated and visualized using the service's HTML interface. To help application developers get started with their projects, CloudThink features a detailed API description together with example queries to its API endpoints. The API itself is straightforward to use for clients: Vehicles are differentiated using their VINs, which clients pass to the API as path parameters directly in the request URL. By making use of query parameters, clients can specify which kind of data they are interested in (e.g., 'SpeedKmHr' or 'Latitude') and can also define boundaries for time-windowing their query. Finally, we give clients the opportunity to define how missing data values should be handled – they can ask the server to either flag missing values, or directly return interpolated values. For example query such as: *https://api.cloud-think.com* accepts the parameters:

- param: The data fields to be fetched, comma-separated;
- start: Start date [yyyyMMddHHmmss];
- end: End date [yyyyMMddHHmmss];
- as media types such as text/html, application/json, application/xml.

Apart from providing access to the cars' data itself, the Gateway Server also manages the bookkeeping of data accesses and registers every access to any bit of stored data. This is necessary to allow tracing of requests for billing purposes and also to accommodate the individual driver's right to know who uses which of their data, and when they do so.

A design choice central to the CloudThink system is to offer maximum support to prospective application developers. For this reason, the Gateway Server also offers a public API where developers may try out their project ideas prior to registering, on a full set of sample data from multiple cars. To help developers get started on their projects, we also provide sample code for multiple platforms (e.g., Android) and in multiple programming languages (e.g., Java, Python, MATLAB). The biggest advantage for a developer when using the Gateway Server as a data broker, though, is that it provides an abstraction from the rather technical OBD data IDs to natural language. By means of this abstraction, developers can, thus, request data for 'SpeedKmHr' or 'Latitude'. A series of database tables manage aliases for vehicular data types, such that new parameters may be added dynamically without interfering with the operation of the Gateway Server or existing applications. Interested developers can register to download technical documentation and open-source code repositories at the project web space.

The API can be accessed here, with sample data being available for testing: *https://api.cloud-think.com/things/1/?parameters=latitude,longitude&startTime=20150101&endTime=20150302*. Sample user credentials for log-in are: 'testApp1' with the password 'test'.

The CloudThink server architecture is deployed on nodes within an elastic computing cloud. The decision to utilize such infrastructure stemmed from the need for scalable infrastructure capable of keeping up with the demands of a large number of connected vehicles, simultaneous application queries, and an increasing volume of data collected. By deploying the CloudThink software on privately managed cloud servers, the appropriate computational power may be allocated dynamically via a third party's management system, allowing server resources to be scaled so as to minimize cost while ensuring a minimum service level. Operation on virtualized machines simplifies software migration to a new physical instance, facilitates hardware replacement and upgrades with minimal downtime, and allows access to highly scalable resources for data storage, processing, and memory. As the platform scales and resources become constrained, it is possible to allocate server tasks increased power, eventually splitting the various tasks to separate servers or even splitting individual tasks across server clusters.

Designing the CloudThink software platform for scalable infrastructure is an architecture choice that requires that the software developed is easy to deploy and optimized to take advantage of a wide range of computational resources. This focused development effort ensures that the CloudThink server software will be able to scale up with computational resources to meet the demands of large numbers of applications and connected vehicles.

Data synchronization. In order to serve API queries with meaningful data, some synchronization methods must be applied to convert serial CAN data to consolidated distinct states of the vehicle. After persisting incoming raw data (see Fig. 5, box 'Persisting'), the CloudThink data server assigns consistent timestamps to the data points in a data cleaning step: for each TCP connection, it attempts to find the earliest data point with a valid UTC timestamp from the GPS transceiver. If no such data point is found, the synchronization fails for all data transmitted within this connection. If multiple UTC time data points are detected for a connection, the server takes the one with the lowest hardware timestamp as an absolute time reference. For each data point in the transmission, it then uses the millisecond offset from that reference point to assign an absolute timestamp. The default assumption is made that data values recorded within less than one second of each other belong to the same vehicle state.

1.4. User Dashboard and Management Back End

The front end for drivers shown in Fig. 6 is being extended and refined to include an application store with the ability to add applications as widgets. Due to the standard Web interfaces used to expose car data, existing applications can easily be ported to the CloudThink platform and offered via the CloudThink store.

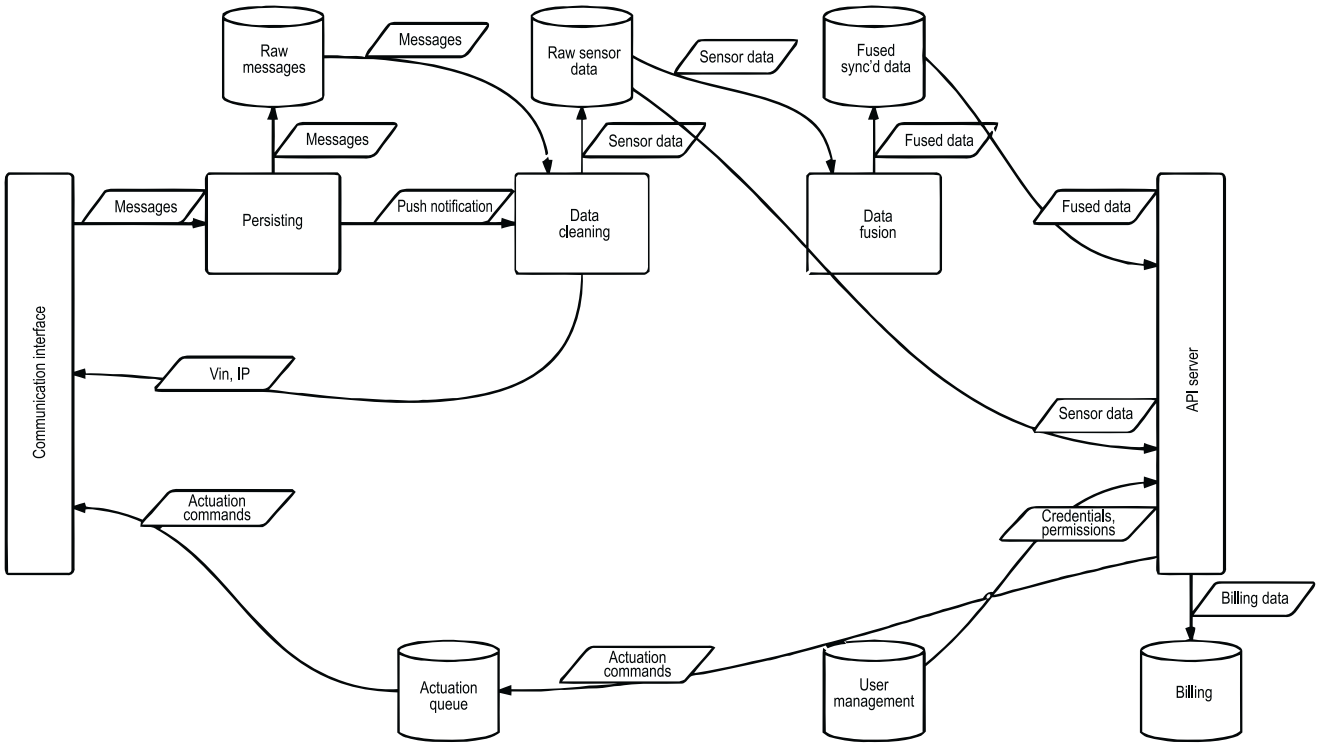

Fig. 5. Overview of the processing steps applied to incoming data packets by the CloudThink Data Server: cylinder is DB storage; square is a server; parallelogram is a process operation

Fig. 6. Screenshot of the prototype CloudThink store for registration and application management

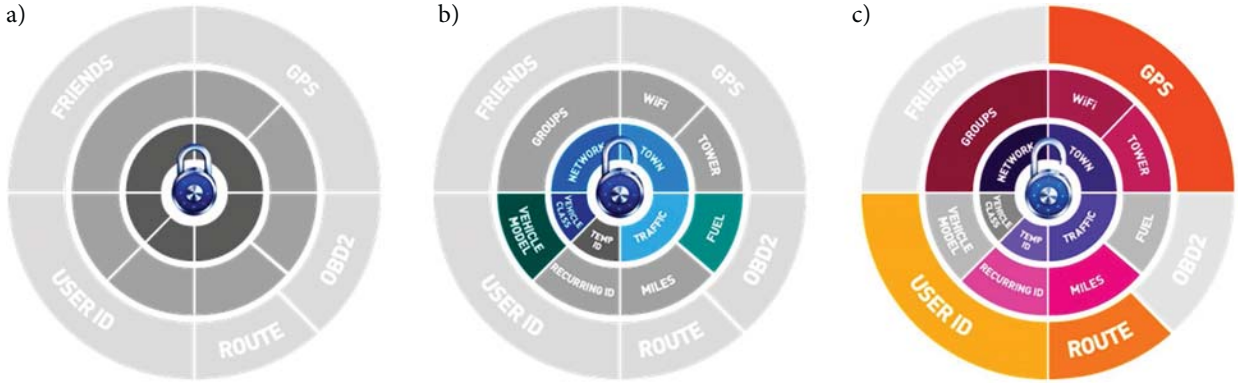

Fig. 7. Security interface for: a – an application for which a user has not released any data; b – a sample 'cheap fuel finder' application without navigation assistance where only some basic location information is exposed; c – a sample 'green navigation with car-sharing' application where much more information has been made available

The unique personal data exposure interface available to CloudThink users shown in Fig. 7 enables a rapid overview of which data may be exposed for various sample applications. Data security was identified as one of the primary concerns for users of location-based services (FCC 2012), and CloudThink uses extensive logging to additionally provide detailed descriptions of how much of each data type was used for various applications.

The task of coordinating permissions for various applications across a fleet of hardware devices running various versions of the firmware to accommodate the different needs of the end users and application builders is challenging. The CloudThink research group has prototyped a system at *http://manage.cloud-think.com*, where users can register, be issued hardware, and begin managing their applications. The important development in this management portal is the ability for system administrators to review and approve the applications submitted by 3rd party authors, and to monitor fair use of data. These aspects have been prototyped in the management back-end as well.

2. Illustrative Applications

The domains of applications enabled by the CloudThink platform cover almost all aspects of transportation energy, environmental, and policy tools, so only a partial list of potential tasks is presented. Many of these applications have already been commercialized by automobile manufacturers and others in insular 'walled garden' solutions without sharing across company boundaries. CloudThink reduces overhead for existing companies as well as aspiring application builders by allowing the driver to select which application he or she wishes to share data. To illustrate how CloudThink data can be used by third-party applications, three use cases are discussed in this section.

2.1. Remote Vehicle Actuation

Successful adoption relies on consumer demand, and as such, ease-of-use features help expand the platform install base. As an example application that drivers would find compelling as a reason to use the CloudThink platform is remote control lock/unlock actuation. Remote locking and unlocking is easily handled by secondary CAN interactions, where a user interacts with an end-use device sending a command and authentication data to the CloudThink server. When the device next uploads data, it receives a response message containing command information and performs the requested actuation, provided the authentication validates successfully. This 'pull' system makes spoofing commands more difficult than a 'push' implementation, as all commands must originate from the target server and be transmitted through the existing socketed connection. A disadvantage is latency, though in practice remote actuation has a short time constant on the order of three to five seconds. These features and more endear the CloudThink platform to consumer users, by providing improved vehicle comfort and convenience (Fig. 8). They are also utilized within a project by members of the CloudThink research team that aims at facilitating the interaction with vehicles using mobile interfaces and object-recognition technology (Mayer, Siegel 2015).

2.2. Vehicle State of Health Monitoring

Health monitoring is another use case uniquely enabled and enhanced by CloudThink vehicle data. With central data storage and historic information as a basis

Fig. 8. Screenshot of the CloudThink remote actuation application, with live demo available here (CarKnow 2014)

for analytics, this same mobile connected vehicle device platform may be used to view and process information for vehicle prognostics, such as monitoring oil performance degradation over time or identifying tire pressure changes prior to a severe loss of pressure taking place (Siegel *et al.* 2014). Vehicle sensors included as part of the On-Board Diagnostic specification may be used to indirectly measure operational parameters, while dedicated, manufacturer-proprietary sensors can provide targeted insights into particular subsystems. Relatedly, sensor data from the CloudThink enabled hardware or collected by a related mobile application allow high-frequency accelerometer analytics, including imbalance identification. Beyond monitoring the characteristics of a particular vehicle instantaneously and over time, parameters may be compared across vehicle makes and models, identifying large-scale problems that might be indicative of a systemic problem as opposed to a wear-based failure.

With this platform installed in vehicles, users may then leverage the data logging function of the hardware to share information with third parties. Insurers, for example, will be interested in using CloudThink data to assess driver behaviour and reconstruct incidents, while government may use related vehicle data to sense the vehicle environment and study infrastructure well beyond the individual vehicle instrumented.

2.3. Fleet Eco-Driving Use Case

The goal of this use case is to investigate whether developing simple cues which drivers of fleet vehicles can follow to adjust their speed up or down (within speed limits) could result in lower fuel consumption. Optimal gear selection in internal combustion engine vehicles is a well-studied problem (Kim *et al.* 2007), but what is less clearly understood is if drivers can speed up or down to augment the transmission's ability to put the engine in a higher efficiency mode of operation. Many factors affect a vehicle's fuel consumption. Theoretically, this depends on (1) the forces the vehicle has to overcome to provide motion rolling, aerodynamic, road grade resistance, and (2) vehicle efficiency, which can vary depending on driving conditions. In practice, for a given vehicle model, these factors can be categorized as follows:

- *vehicle-related:* its payload, use of air conditioning and other auxiliary loads, equipment that may influence its aerodynamics, e.g. roof racks, maintenance schedule, age;
- *environment-related:* ambient temperature, humidity and precipitation;
- *fuel-related:* fuel type used; and
- *travel-related:* road grade, drive cycle, which depends on chosen route, traffic conditions, and driver style.

Focusing on driving style, fuel consumption is known to be lowest during steady-speed driving at moderate speed, and it increases almost linearly with acceleration (Jones 1980). For transient driving, Berry (2010) has found that the impact of aggressive driving behavior on fuel consumption depends on the travelling speed bands. For low speed/neighborhood driving (<32 kph) and high speed/highway driving (>72 kph), the average velocity dominates the impact on fuel consumption. During moderate speed/city driving (32–72 kph), accelerations have greater effect on fuel consumption.

In order to provide effective speed signals to drivers, an understanding of the vehicles transmission gear ratio must be obtained. This is relatively easily accomplished, by comparing velocity of the vehicle to its engine's angular velocity measured in RPM and making an assumption about the tire radius. The number of clusters to fit is obtained from the CloudThink database for vehicle transmissions, derived by querying an API opened by the Edmunds Automotive Review as soon as a new VIN is detected. Fig. 9 shows the resulting identification for a test vehicle driven over 10000 km.

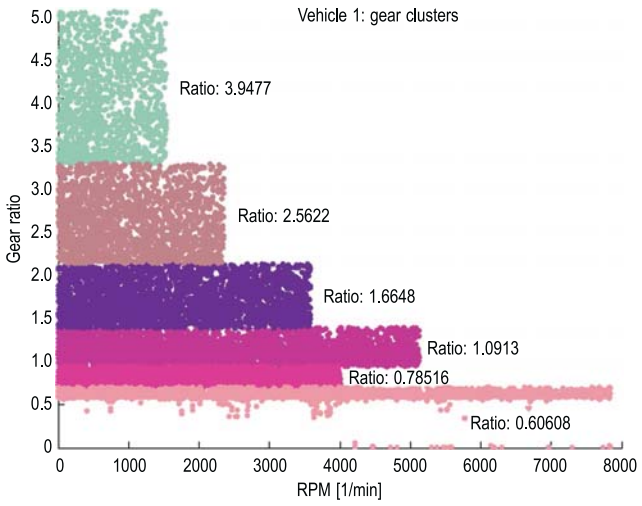

Fig. 9. Identified gear ratios using *k*-means using the cityblock clustering distance agree well with true gear ratios

The true gear ratios are 4.14/2.37/1.55/1.15/0.86/0.68 and the identified ratios are 3.95/2.56/1.66/1.09/0.79/0.61, a very respectable agreement.

Conclusions

The use cases presented leverage vehicle data to improve vehicle performance, reliability, efficiency, and user experience. CloudThink is an enabling platform that facilitates the collection and utilization of these data, and reduces development complexities ensuring these and other applications can be built and iterated upon quickly.

The use-cases have mostly varying data requirements, which illustrates nicely the importance of the principles of flexibility and security upon which the CloudThink platform is built.

The CloudThink platform has been designed and engineered to strike a balance between the flexibility to host many applications with the ability to securely and privately store user data. This careful navigation of the trade-off between security and flexibility is the primary differentiator between CloudThink and the myriad of competing telematics platforms available worldwide.

The goal of this work is to develop a system which will become a de-facto standard for projecting vehicle data into the cloud, and for enabling third-party applications to access data which drivers have authorized them to access, via an easy-to-use API.

CloudThink aims to be a fair, unbiased broker of vehicle data.

Acknowledgements

The authors would like to acknowledge the support of the SUTD-MIT International Design Centre in both funding as well as supporting the development of the CloudThink standard through its association with the various IDC design projects in the transportation systems arena. We furthermore thank the Swiss National Science Foundation (Grant Number 134631).

References

Berry, I. M. 2010. *The Effects of Driving Style and Vehicle Performance on the Real-World Fuel Consumption of U.S. Light-Duty Vehicles*: MSc Thesis. Massachusetts Institute of Technology. 140 p. Available from Internet: http://web.mit.edu/sloan-auto-lab/research/beforeh2/files/IreneBerry_Thesis_February2010.pdf

CarKnow. 2014. *CarKnow LLC: Media*. Available from Internet: http://www.carknow.me/media

Duri, S.; Elliott, J.; Gruteser, M.; Liu, X.; Moskowitz, P.; Perez, R.; Singh, M.; Tang, J.-M. 2004. Data protection and data sharing in telematics, *Mobile Networks and Applications* 9(6): 693–701. http://dx.doi.org/10.1023/B:MONE.0000042507.74516.00

Duri, S.; Gruteser, M.; Liu, X.; Moskowitz, P.; Perez, R.; Singh, M.; Tang, J.-M. 2002. Framework for security and privacy in automotive telematics, in *WMC'02: Proceedings of the 2nd International Workshop on Mobile Commerce*, 28 September 2002, Atlanta, GA, United States, 25–32. http://dx.doi.org/10.1145/570705.570711

Evans, D. 2011. *The Internet of Things [INFOGRAPHIC]*. Available from Internet: http://blogs.cisco.com/diversity/the-internet-of-things-infographic

FCC. 2012. *Location-Based Services: an Overview of Opportunities and other Considerations*. Federal Communications Commission (FCC), Washington, DC. 46 p. Available from Internet: https://www.fcc.gov/document/location-based-services-report

Iqbal, M. U.; Lim, S. 2010. Privacy implications of automated GPS tracking and profiling, *IEEE Technology and Society Magazine* 29(2): 39–46. http://dx.doi.org/10.1109/MTS.2010.937031

ISO 15031-5:2015. *Road Vehicles – Communication Between Vehicle and External Equipment for Emissions-Related Diagnostics – Part 5: Emissions-Related Diagnostic Services*.

ISO 15765-2:2011. *Road Vehicles – Diagnostic Communication over Controller Area Network (DoCAN) – Part 2: Transport Protocol and Network Layer Services*.

ISO 15765-4:2011. *Road Vehicles – Diagnostic Communication over Controller Area Network (DoCAN) – Part 4: Requirements for Emissions-Related Systems*.

Jones, R. 1980. *Quantitative Effects of Acceleration Rate on Fuel Consumption*. Technical Report No. EPA=AA-SDSB-80-06. US Environmental Protection Agency (EPA). 23 p.

Gerardo, B. D.; Lee, J. 2009. A framework for discovering relevant patterns using aggregation and intelligent data mining agents in telematics systems, *Telematics and Informatics* 26(4): 343–352. http://dx.doi.org/10.1016/j.tele.2008.05.003

González, M. C.; Hidalgo, C. A.; Barabási, A. L. 2008. Understanding individual human mobility patterns, *Nature* 453: 779–782. http://dx.doi.org/10.1038/nature06958

Guinard, D.; Ion, I.; Mayer, S. 2011. In search of an internet of things service architecture: REST or WS-*? A Developers' Perspective, *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* 104: 326–337. http://dx.doi.org/10.1007/978-3-642-30973-1_32

Kim, D.; Peng, H.; Bai, S.; Maguire, J. M. 2007. Control of integrated powertrain with electronic throttle and automatic transmission, *IEEE Transactions on Control Systems Technology* 15(3): 474–482. http://dx.doi.org/10.1109/TCST.2007.894641

Litman, T. 1997. Distance-based vehicle insurance as a TDM strategy, *Transportation Quarterly* 51(3): 119–137.

Mayer, S.; Siegel, J. 2015. Conversations with connected vehicles, in *IoT 2015*: *Proceedings of the 5th International Conference on the Internet of Things*, 26–28 October 2015, Seoul, Korea, 1–7.

Musicant, O.; Bar-Gera, H.; Schechtman, E. 2010. Electronic records of undesirable driving events, *Transportation Research Part F: Traffic Psychology and Behaviour* 13(2): 71–79. http://dx.doi.org/10.1016/j.trf.2009.11.001

Rogers, S.; Langley, P.; Wilson, C. 1999. Mining GPS data to augment road models, in *KDD'99: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 15–18 August 1999, San Diego, CA, United States, 104–113. http://dx.doi.org/10.1145/312129.312208

SAE. 2010. *On-Board Diagnostics for Light and Medium Duty Vehicles Standards Manual – 2010 Edition*. Published by SAE International with a Product Code of HS-3000/2010.

Sarma, S. 2004. Integrating RFID, *Queue* 2(7): 50–57. http://dx.doi.org/10.1145/1035594.1035620

Shi, W.; Liu, Y. 2010. Real-time urban traffic monitoring with global positioning system-equipped vehicles, *IET Intelligent Transport Systems* 4(2): 113–120. http://dx.doi.org/10.1049/iet-its.2009.0053

Siegel, J.; Bhattacharyya, R.; Deshpande, A.; Sarma, S. 2014. Vehicular engine oil service life characterization using onboard diagnostic (OBD) sensor data, in *2014 IEEE SENSORS: Proceedings of the International Conference*, 2–5 November 2014, Valencia, Spain, 1722–1725. http://dx.doi.org/10.1109/ICSENS.2014.6985355

Smith, R.; Shahidinejad, S.; Blair, D., Bibeau, E. L. 2011. Characterization of urban commuter driving profiles to optimize battery size in light-duty plug-in electric vehicles, *Transportation Research Part D: Transport and Environment* 16(3): 218–224. http://dx.doi.org/10.1016/j.trd.2010.09.001

Tahat, A.; Said, A.; Jaouni, F.; Qadamani, W. 2012. Android-based universal vehicle diagnostic and tracking system, in *2012 IEEE 16th International Symposium on Consumer Electronics (ISCE)*, 4–6 June 2012, Harrisburg, PA, 137–143. http://dx.doi.org/10.1109/ISCE.2012.6305105

Toledo, T.; Lotan, T. 2006. In-vehicle data recorder for evaluation of driving behavior and safety, *Transportation Research Record: Journal of the Transportation Research Board* 1953: 112–119. http://dx.doi.org/10.3141/1953-13