Context-Aware Middleware for Vehicular Applications

Jian Wang, Weiwen Deng and Peng Zhou

Abstract Developing vehicular distributed applications faces many challenges because most of them apply their specialized communication protocols and technical standards. We propose Context-Aware Middleware for Vehicular Applications (CAMVA), which can react to around environments adaptively and timely. CAMVA uses components based design pattern, and is optimized a lot in terms of vehicular complexity and special requirements of vehicular applications on security and immediacy, by which software programmers can develop and deploy vehicular applications quickly and reliably through assembling, plugging and articulating the existing components even though they are not familiar with the bottom details. CAMVA is located between application level and operation system level, and is composed of collection layer, core layer, running layer, component container, and component library. CAMVA realizes context-aware ability and supports complex vehicular environments. CAMVA behaves better in immediacy, expansibility, static configurability, and dynamic adaptability aspects, so it can achieve strict requirements of intelligent vehicles on middleware.

Keywords Middleware · Vehicular applications · Context-aware · Vehicular communications · Data collection

F2012-D03-012

J. Wang (✉) · P. Zhou
College of Computer Science and Technology, Jilin University, Changchun 130012, China
e-mail: wangjian591@gmail.com

W. Deng
State Key Laboratory of Automotive Simulation and Control, Jilin University, Changchun 130012, China

SAE-China and FISITA (eds.), *Proceedings of the FISITA 2012 World Automotive Congress*, Lecture Notes in Electrical Engineering 194,
DOI: 10.1007/978-3-642-33829-8_40,

1 Introduction

With the quick development of vehicular electronic and vehicular communications, the approach to efficiently develop and deploy the distributed applications in vehicle-to-vehicle situation attracts more and more attention [1]. Middleware can realize the interoperation between heterogeneous systems [2], abstract and encapsulate the universal processes into a uniform supportive layer, such as network communication, security and transaction services [3–5]. Vehicular networks have many different characteristics from the wired works, such as limited bandwidth, frequently changed topology and high requirements on security. Herein the middleware applied in vehicular networks has to meet the following requirements: 1. Instantaneity. Most the safety applications exert rigorous requirements on time delay. 2. Stability. The quick moved and changed vehicular driving environment demands that middleware should behave better stably and invulnerably. 3. Adaptivity. The frequently changed driving situation also requires that middleware should dynamically change configurations in order to be adaptive to surround changes.

We design and implement a vehicular-network-oriented middleware CARMVE (Context-Aware Reflective Middleware in Vehicular Environment). CARMVE uses multi-component structure and markup language based synchronization protocol, in order to improve real-time and stability, and then to achieve the requirements of intelligent vehicles on middleware system.

2 System Architecture

Figure 1 gives the system architecture of CARMVE which locates between application layer and operation system layer. It is composed of collection layer, core layer, running layer, component container, and component library.

Collection layer is to collect and measure the current states and parameters, such as network state, device running state, users' profile, software state, and surround environment state. It is composed of many separated collection components.

Core layer is the key part of CARMVE. It is composed of script parser, adaptivity management and context-aware management. Script parser initializes the application configuration by parsing and analysing application configuration script defined by XML language. The application configuration script is exampled in Fig. 2. The label component lists all components used in applications, and the label rule gives all the rules. Context-aware management is responsible for organizing and computing the context information collected by collection layer, and then decide whether a adaptivity action is launched or not. After a adaptivity action is launched, adaptivity management searches the corresponding component instance or application in running layer according to user-defined adaptivity rules, and attempts to finish adaptivity process by modifying attribute values and changing behavioral structures.

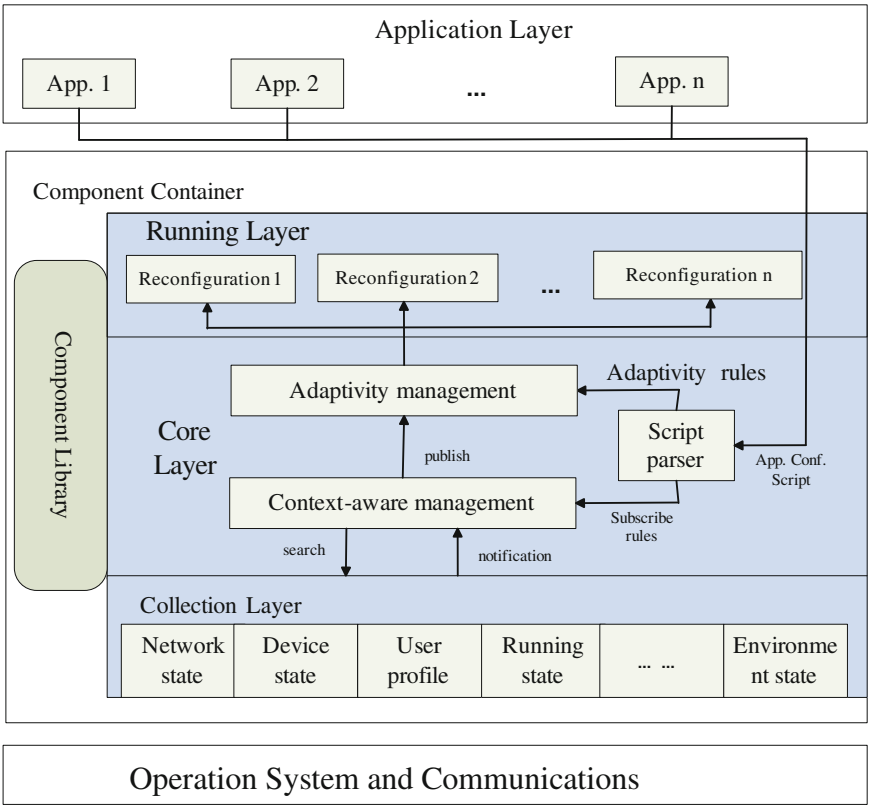


Fig. 1 System architecture of CAEMVE

Fig. 2 The configuration script

```
<Components>
        <component> </component>
    …
</Components>
<Rules>
     <rule>
                    < Trigger ></ Trigger >
                    <Componentchain></Componentchain>
        </rule>
        …
 </Rules>
```

Running layer provides a reconfiguration to every application. The reconfiguration encapsulates the necessary component instances and some component chains which correspond to adaptivity rules. Only one component chain stays in active state at the same time. The reconfiguration realizes the adaptivity process by

changing the attribute value of component instance or activating the proper component chain.

Component container is a special part in CARMVE. All the applications based on CARMVE are running as a component chain in the component container. Component container provides running environment to other CARMVE component, receives the registration and cancelation of components, publishes the component services, and so on.

Component library records every registered component. The developers could resemble the existing components and then generate a highly efficient and high-quality vehicular distributed software system.

In practical, the subscribed adaptivity actions are always not provided only by a single context-aware component, but by more than one context information. For example, in automatic tracing system, when the velocity and distance meet a certain relation, the system needs to adapt acceleration, deceleration, stop, launch, and others. Herein, script parser generates a transaction tree for every application configuration script. Figure 3 shows the example of transaction tree. Context-aware management executes the realized subscribe and publish according to the transaction tree.

3 System Instance

We explore the role of CARMVE in vehicular communications by using the case showed in Fig. 4.

Vehicle A and vehicle B are driven in the same direction, vehicle A is in ahead, and vehicle B is in behind. At one time, vehicle A finds some dangerous areas in ahead, but because of blocked sight, vehicle B can't detect the danger. So at this time, vehicle B may discover the potential danger by using the image data transmitted from vehicle A. However, the network environment between two vehicles is dynamically changed, such as bandwidth, so the transmission process needs to be changed in real time according to network situation, such as compressing ratio for real-time and clarity during image transmission. The applications based on CARMVE could solve this problem well.

Suppose the bandwidth between vehicles A and B is less than 2M, the image compress is launched, and the compressing ratio is 50 %. The configuration script of the application is as follows:

<Components>
<component>
<name> GRAB </name>
…
</component>
<component>
<name > COMPRESS </name> component>

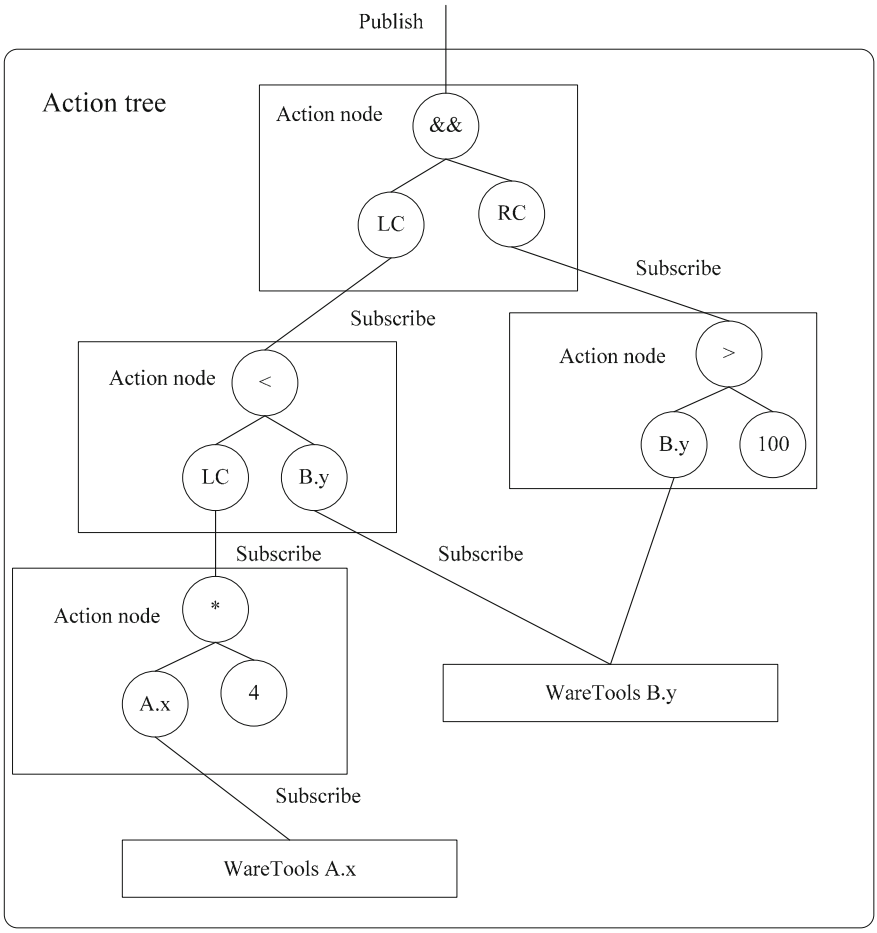
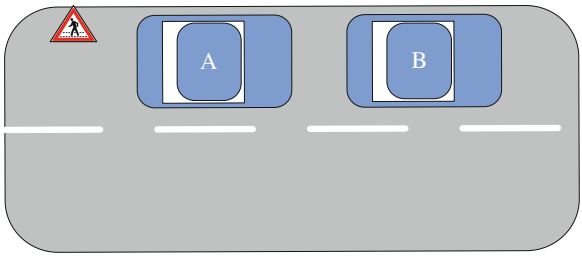


Fig. 3 CARMVE context-aware model

Fig. 4 An example for vehicular applications based on CARMVE



...

```
</component>
<component>
<name> DECOMPRESS </name>
…
</component>
<component>
<name> DISPLAY </name>
…
</component>
</Components>
<Rules>
<rule>
<Trigger>
<event>
<Operator> LT </Operator>
<LC>
<Output> networkawareness.bw </Output>
</LC>
<RC>
<const> 2 </const>
</RC>
</event>
</Trigger>
<Componentchain type="Send_Chain">
<SetParam>
COMPRESS.CompressQuality=50
</SetParam>
<SetChain>
GRAB.PtrOutput -> COMPRESS.PtrInput;
COMPRESS.StreamOutput -> SEND;
Grab.Start
</SetChain>
</Componentchain>
<Componentchain type="Re_Chain">
<SetChain>
RECEIVE -> DECOMPRESS.StreamInput;
DECOMPRESS.StreamOutput -> DISPLAY.Input;
</SetChain>
</Componentchain>
</rule>
…
</Rules>
```

We declare four components in the script: WebCam video collector, JPEG-Compress compress component, JPEGDeCompress decompress component, and display component.

4 Conclusion

CARMVE decreases developing cost and complexity of vehicular applications, and improves flexibility, adaptability, and portability. It brings theory and practical effects on vehicular applications.

References

1. Wong KD, Tepe K, Chen W et al. (2006) Inter-vehicular communications. IEEE Wirel Commun 13(5): 6–7
2. Yang DR, Luan J, Jun-zhong GU (2005) Comparative research on middleware techniques. Appl Comput Syst (3):27–30
3. Ye F, Adams M, Roy S (2008) V2V wireless communication protocol for rear-end collision avoidance on highways. In: Proceedings of the IEEE communications workshops. Piscataway, IEEE, pp 375–379
4. Huan Z, Shou-zhi X, Cheng-xia L (2009) A V2V broadcast protocol for chain collision avoidance on highways. In: Proceedings of 2009 IEEE International Conference on Communication Technology and Applications, pp 2062–2067
5. Tang A, Yip A (2010) Collision avoidance timing analysis of DSRC-based vehicles. Accid Anal Prev 42(1):182–195