# Multithreading Embedded Multimedia Application for Vehicle Blackbox

Jajin Koo, Jeongheon Choi, Youjip Won, and Seongjin Lee

**Abstract.** Recent introduction of Vehicle Black Box system enabled the user to use the system to collect driving information such as location, time, and satus of the vehicle, which can be analyzed to find the cause of the accident. In this paper, we provide an efficient way of using the data generated from Vehicle Black Box system. We analyze the performance of proposed multi threaded system by comparing it with single threaded implementation of the system. The paper shows that the processing time, and initialization time of the multi threaded system is 39%, 233% faster than single threaded Vehicle Black Box System, respectively. We also show that response time of multi threaded system for single control message is about 10% faster than single threaded system.

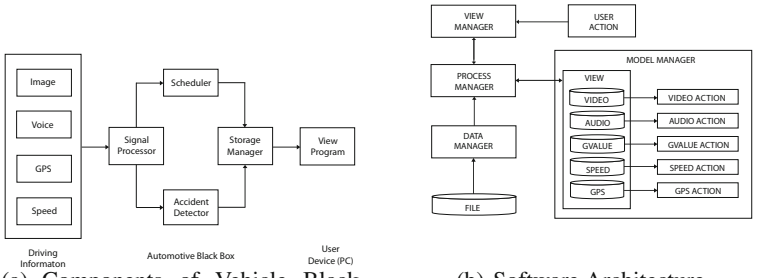
## 1 Introduction

Recent introduction of Vehicle Black Box system enabled the user to use the system as supporting material in the event of vehicle accidents and exploit the system to enhance driving experience. Main purpose of the vehicle black box is to collect driving information such as location, time, and satus of the vehicle, which can be analyzed to find the cause of the accident. Modern systems also capture video and audio data along with other status information, which increases overhead in the embedded system.

Chet et al. [1] designed the black box as warning system to prevent users from exceeding speed limits, and Kassem et al. [2] focused on minimizing the system in hardware level, while maintaing similar level of functionalities as other black box systems. Shui et al. [3] developed black box system based on ARM+DSP to monitor and record environmental information while users drive the vehicle; this system introduces exploiting video recorder on black box system. Some of recent

Jajin Koo · Jeongheon Choi · Youjip Won · Seongjin Lee
Hanyang University, Seoul Korea
e-mail: {space0215,rhythmical,yjwon,james}@ece.hanyang.ac.kr

F.L. Gaol (Ed.): Recent Progress in DEIT, Vol. 1, LNEE 156, pp. 379–384.
springerlink.com 




(a) Components of Vehicle Black Box System

(b) Software Architecture

**Fig. 1** Components and Software Architecture of Vehicle Black Box System

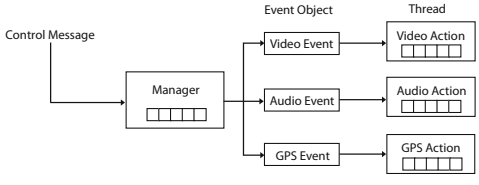



**Fig. 2** Configuration using Multi-Thread Method

work on the field [4, 5] presents multi-threading programming model on distributed environment to enhance the processing speed.

Most of the system disregards the efficiency in manipulatiing the generated data and processing them to form useful information. Typcial black box system captures video data, audio data, pressure, and GPS data. In this paper, we develop a new viewer system that exploits efficient processing techniques to illustrate and present generated data. Another important issue in black box is to utilize resources while enhancing response time and processing time; to provide the remedy in enhancing reponse and processing time, we exploit multi-thread in implementing the system.

## 2 Vehicle Black Box System

Main purpose of Vehicle black box system to capture real-time driving information to provide data which can be used in analyzing cause of accidents. Most existing vehicle black box system employees Digital Video Recorder (DVR) to record scene or same view as the driver. Recent vehicle black box system not only captures the scene, but also captures speed of the vehicle, direction, acceleration and break operation, external pressure, and many more. The black box system exploits vehicle sensors and Global Positioning System (GPS) to collect such data.

Fig. 1(a) illustrates the components used in vehicle black box system with Driving Information, Automotive Black Box, and User Device. Driving Information is considered as generated data while the vehicle is in active mode. It collects images,

voice, GPS, speed through sensors attached to Automotive Black Box. Collected general driving data is stored in Automotive Black Box. It manages data collected by the Driving Information. We separate general driving data and accident driving data according to policy of Storage Manager and saves the data to the external storage device. User Device is a front-end user interface that presents stored Driving Information to the user.

## 3 System Modeling

This paper focuses to efficiently configure Viewer which presents collected data. The Viewer not only generetes appropriate views to the user with the collected data, but also decodes and encodes the video data, reflects user control information, and manipulates GPS information in real-time fashion. In process of generating several views for specific purposes, single thread application cannot efficiently deal with synchronization issues. And, it has to suffer from slow response time when user messages coincide with multiple other system generated messages.

We design Viewer that presents the data collected by black box and Fig 1(b) illustrates the software architecture. First component in the viewer is Data Manager (DM) which processes data; DM loads driving data from storage device (SD card, HDD) and analyzes a file header to distinguish video, audio, GPS, and etc. DM transmits the data to each Viewer. Second is Viewer Manager (VM) which is implemented with Model-View Control (MVC) software design pattern. VM collects control messages from user and acts as a agent and interface between user and program. Process Manager (PM) sends control messagees from VM and parses data collected from DM for each Viewer. Model Manager (MM) creates and manages the Viewer for each data; it also processes control messages issued by user.

## 4 Multi-thread Technique

As Section 3 describes there are synchronization and response time issue in single-threaded implementation of the system. In this paper, we propose multi-threaded black box system that provides remedy to both issues. Fig. 2 illustrates how Manager make use of thread according to different events from three Viewers (video, audio, and GPS data.) When Viewer Manager receives Control message, it calls Event Object that runs on separate thread.

Control message is processing unit which drives each viewer. User requests (play, stop, pause, and etc) and application generated requests (file open, close) are examples of control messages. The Manager is responisble for creating and managing each viewer, and storing the control messages. Both Manager and Viewer thread runs on user level thread and they both keep queue to store received control messages. Viewer threads processes the data which are collected by black box. Event object is kernel mode synchronization object for Manager and Viewer thread. When black box system is activated, it first creates appropriate environment for the system to run thread as shown in Fig. 2. Next, the Manager thread waits for messages

from user or the system itself, and the Viewer thread also waits for messages from the Maanger thread. When the user or the system issues messages, the Manager thread stores messages in the queue, and each Event Object, which is associated each Viewer, is set as signaled. When the Evenet Object is set to signaled, the Manager thread delivers the messages to each Viewer thread, and Viewer thread executes individual requests using each queue message, until all messages from the queue is processed. After all messages processed, the Event Object resets its status to *Not-signaled* and waits until the state changes to *Signaled*.

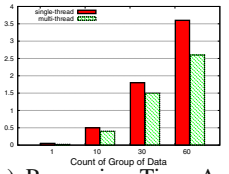
## 5 Experiment

We run and test the black box system under Intel(R) Core 2 Duo E8200 and 2Gbyte of DDR2 Memory under Windows XP 32bit. We develop the Viewer system that presents Vehicle Black Box System data, implemented with MFC Microsoft video C++ 6.0. Since main objective of the paper is to produce a system with better response time, we implemented the system to run on single thread and multi threaded environment. As described in Section 4 the threads used in the system is user level threads provided by the Operating System. In order to handle thread synchronization, we exploit event synchronization.

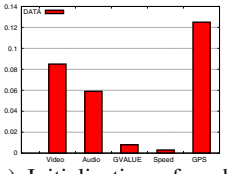
### *5.1 Processing Time*

In order to measure processing time, we compare processing time for given size of data between the single threaded system and multi threaded system. Data used in this experiment is video, audio, GVALUE (pressure of the vehicle), speed, and GPS of the vehicle. A unit of data is denoted as Pack. The Pack consists of one minute of video data, one minute of audio data, and sixty consecutive GVALUE, Speed, and GPS which are collected every second. In order to measure the processing time, we introduce another unit called Group of Data (GoD). For example, GoD of one corresponds to ten frames of video data, ten frames duration of audio data, single GVALUE, single Speed, and single GPS data.

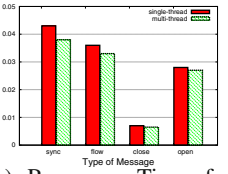
We range the GoD value from 1 to 60 and measure the response time and measure the response time of single threaded system and multi threaded system. Fig. 3(a) illustrates the difference between the two system. In case of the single threaded system, Viewer processes video, audio, GVALUE, Speed, and GPS information sequential fashion. In other words, FFMPEG decodes video and audio data; then Viewer draws GVALUE and Speed graph; finally, GPS information is processed. Under such contraints, each Viewers has to wait data processing time, I/O delays, and Memory access delays. When the system is running under multi thread environment, independent run of each thread removes processing delays. As number of GoD increase the effect becomes clear; when the number of GoD is 60 meaning 600 frames of video data, multi threded system is about 39% faster than single threaded system.




(a) Processing Time According Count of GoD




(b) Initialization of each viewer




(c) Response Time for Control Message

**Fig. 3** Processing Time and Initialization Time

## *5.2 Initialization Time*

Next, we measure the initialization time of the two systems. It is important to reduce the initialization time as it decides the user experience. It accounts of time the system is loaded and seting the Event Object to *Not-Signaled* state. It excludes the time spent in loading metadata from a file to memory, and separating the data corresponding to each Viewers. Assuming there are loaded data from files on each Viewer, initialization time measures the neccesary time for memory allocation, time spent in data parsing. Fig. 3(b) illustrates initialization time for each Viewer. GPS Viewer uses about 0.12 seconds, Video Viewer consumes about 0.08 seconds, and Audio Viewer consumes about 0.06 second in initializing the Viewer. Unlike GPS, Video, Audio Viewers, which has to initialize different codecs and connect to GPS server, GVALUE and Speed only has to retreive data from the sensor in Automotive Black Box in Fig. 1(a). GVALUE and Speed Viewer consumes 6.5msec and 0.3msec, respectively.

We meausred the sum of initialization time difference between the two systems. It shows that multi threaded system (120 msec) is 233% times faster than single threaded system (280 msec). Note that the system initialization time is same as longest time consuming operation, which is GPS Viewer, since all other Viewers has to wait until GPS Viewer is ready to synchronize with other data on the Viewers.

## *5.3 Response Time*

Third experiment measures the response time for external control messages of single threaded system and multi threaded system. The control messages used in the experiment is as follows: SYNC (Move position), FLOW (Synchronization message), CLOSE (Exit the program), and OPEN (Start the program). Fig. 3(c) illustrates the difference between the two systems. We measure the response time of each control messages. Similar to processing time experiment (Fig. 3(a)) multi threaded system shows better response time than single threaded system.

The response time of SYNC message, which updates position information, is reduced about 100 msec; FLOW message, which sychronizes the data between each viewer, is reduced about 2 msec. Response time for open and close message is almost same for both systems but slighter shorter for the multi threaded system. The

result shows that effect of multi threading in response time is not too great compared to processing time. Note that the result of Fig. 3(c) is response time of single message. Therefore, effect of multi threading in the real system can be singnificant when there are multiple messages. Proposed system successfully addresses the issue in response time in handling control message and the system resource utilization.

## 6 Conclusion

As the data generated by the embedded system is increasing, the system has to efficiently manage the utilization resources. Vehicle Black Box system is one of such area, which requires to process large amount of data. In this paper, we provide an efficient way to manipulate and manage various data in a single system with short processing and response time. We propose multi threaded system that independently processes data through separate data veiwer. We analyze the performance of proposed system by comparing it with single threaded implementation of the system. The paper shows that the processing time, and initialization time of the multi threaded system is 39%, 233% faster than single threaded Vehicle Black Box System, respectively. We also show that response time of multi threaded system for single control message is about 10% faster than single threaded system.

**Acknowledgements.** This work was supported by Mando Corporation and the NRF (National Research Foundation) grant funded by the Korea government (MEST) (No.R0A-2009-0083128).

## References

1. Chet, N.C.: Design of black box for moving vehicle warning system. In: Proceedings. Student Conference on Research and Development, SCORED 2003, pp. 193–196 (2003)
2. Kassem, A., Jabr, R., Salamouni, G., Maalouf, Z.: Vehicle black box system. In: 2008 2nd Annual IEEE Systems Conference, pp. 1–6 (2008)
3. Shui, Y., Zhen, D., Hong-Bin, C., Jin-Zhong, C., Lei-Ting, C.: Design of black box of vehicle system based on arm+dsp. In: Application Research of Computers 2008 (February 2008)
4. Tian, L., Ming Fan, S., Zhao, W.: Application and research of multi-thread technology in spaceborne sar imaging. In: International Conference on Space Information Technology (2010)
5. Qin, Z., Gu, J.: Multi-thread technology based autonomous underwater vehicle. In: 2010 8th IEEE International Conference on Control and Automation (ICCA), pp. 898–903 (2010)