Prototyping Commodity ICT for Mobility CPS

Yuki Iida‡, Yusuke Fujii‡, Takuya Azumi‡, Nobuhiko Nishio‡,
Shinpei Kato*, Tsuyoshi Hamada§, Satoshi Kagami¶, Nobuo Kawaguchi†, and Kazuya Takeda*

* Graduate School of Information Science, Nagoya University
† Graduate School of Engineering, Nagoya University
‡ College of Information Science and Engineering, Ritsumeikan University
§ Nagasaki Advanced Computing Center, Nagasaki University
¶ Digital Human Research Center, AIST

***Abstract*—The next-generation mobility infrastructure requires computer systems to be more and more intelligent and interactive with the physical world. A grand challenge to this cyber-physical systems (CPS) problem can be found in real-time processing of autonomic control and environmental perception. Given the large volume of information in the physical world, the computational cost of autonomic control and environmental perception is likely very high, which may not be affordable for embedded mobile devices. In this paper, we explore a possibility of leveraging cloud computing technology to support such computationally expensive operations of intelligent mobility CPS applications. Specifically we quantify the overhead of offloading computations to the cloud using commodity information and communication technology (ICT) platforms taking an example of autonomous driving. This quantification of overhead plays a vital role in the design issue of emerging mobility CPS applications. Our experimental results demonstrate that the current communication devices including WiFi and LTE achieve sufficient throughput and latency even when used with smartphones to transfer large images and high-rate control commands over the network.**

***Index Terms*—Cloud Computing; Smart Devices; CPS**

I. Introduction

Mobility is an essential piece of our life. In recent years, transportation systems and automobiles are becoming more and more intelligent underlying high-efficient mobility of the societal infrastructure. Such innovations in mobility could reduce car accidents, remove humanś stress, improve traffic throughput, and create new markets. Towards this end, the next-generation mobility infrastructure is expected to achieve a tight coordination of computer systems and physical elements, often referred to as cyber-physical systems (CPS), facilitating a cyber-understanding of the physical world. A core challenge to this mobility CPS, however, is a power and space constraint of each mobility component. For example, moving vehicles cannot accommodate rich computer systems due to the limited battery power. Particularly the computational cost with respect to a cyber-understanding of the physical world is very high and the computing capability of current vehicular embedded systems is woefully inadequate. In order to address this trade-off, mobility CPS must seek for a cooperation with advanced information and communication technology (ICT) such as cloud computing and smart devices.

A good example of computationally-expensive mobility CPS applications is an autonomous vehicle [1], [5], [8], which is required to recognize roads, traffic signs, surrounding vehicles, and pedestrians in real-time. An autonomous vehicle in the current state of the art tends to use laser sensors and/or cameras for those environmental perception tasks. The laser sensors can detect object edges as a set of 3-D points by hardware, reducing the computational requirement imposed on the vehicular system software, but are generally very expensive way beyond consumer electronics prices. On the other hand, the cameras are less expensive in price but come available in exchange with computational cost, because image processing is highly compute-intensive. It would require a rich set of multicore CPUs and hardware accelerators such as GPUs to meet the desired frame rate. Unfortunately these devices may not be affordable for battery-operated vehicular systems due to power consumption issues. As aforementioned, therefore, we should seek for a possibility of employing cloud computing technology to offload compute-intensive tasks onto high-performance computing (HPC) servers over the network. A question raised herein is "what is the overhead of offloading computation and associated data to the cloud?" If this overhead is acceptable, commodity ICT platforms will be a strong basis for mobility CPS applications.

Contribution: We present a prototype implementation of commodity ICT platforms for mobility CPS applications to quantify the overhead of offloading computation and data to the cloud. Specifically we focus on an autonomous vehicle as an example of mobility CPS applications, and leverage a commodity smartphone and PC server to process images and control the autonomous vehicle. The overhead of transferring images over the network governs the frame rate of image processing in the cloud while that of transferring control commands determines the minimum feedback-control period of autonomous driving. We demonstrate that the bottleneck of networked image processing can be found in the computation time itself rather than the network communication overhead. We also find that the network throughput of commodity ICT is sufficient to execute autonomic control but the worst-case latency must be bounded to provide stability. Our contribution is a useful insight into a coordination of commodity ICT and mobility CPS.

Organization: The rest of this paper is organized as follows. Section II describes the basic concept behind this

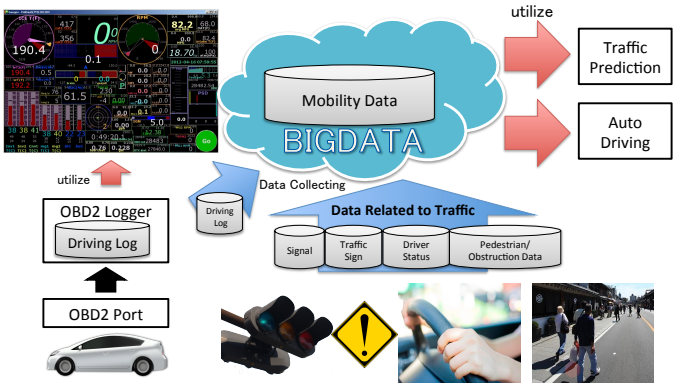

Fig. 1. Collecting automotive and environmental data.

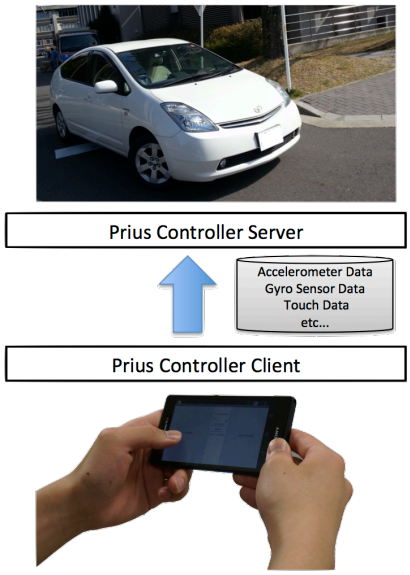

Fig. 2. A smartphone application for remote vehicle control.

paper. Section III presents our prototyping of ICT platforms for an autonomous vehicle as an example of mobility CPS applications. Section IV provides the evaluation of overhead imposed on data transfers over the network relevant to remote vehicle control and networked image processing. This paper concludes in Section V.

II. Basic Concept

We advocate a concept of "mobility CPS in the cloud". Many mobility CPS applications are often battery-operated, and cannot accommodate large power consumers unless they equip special battery facilities. Integrating cloud technology, however, allows computations and data to be offloaded over the network and we are freed from power consumption issues. For simplicity of description, this paper presents our concept in the context of autonomous driving, but it is highly applicable for many other mobility CPS applications.

Grand challenges of mobility CPS include a modeling of the physical world that underlies a real-time cyber-understanding of the physical world including environmental perception and motion control. This modeling of the physical world requires accumulative collection of real-world data, often referred to as "Big Data". Fig. 1 illustrates an example of collecting automotive data through the on-board diagnosis (OBD) connectors as well as environmental data. We are building this system using commodity ICT products. The automotive data can be obtained through the CAN bus, while the environmental data can be captured using mobile devices such as smartphones. These data sets can actually be shared with a lot of mobility CPS agents. Such "Big Data" trends also encourage the concept of mobility CPS in the cloud. Since the data sets are stored in the cloud, each mobility CPS agent needs to access the network. For example, we can store a very large global trained data set [7] for anonymous agents to perform image recognition. The first step towards this approach is to understand the scale of latency and overhead imposed on cloud computing in real-time.

III. Prototyping

In this paper, we provide a prototyping of ICT platforms for mobility CPS applications, particularly taking an example of autonomous driving. We apply the cloud computing paradigm for autonomous driving to complement wimpy vehicular embedded systems. One of the major goals of this cloud-based autonomous driving system is to offload the computational requirement of autonomic control and environmental perception to remote HPC servers from local vehicular systems. We focus on ICT platforms. Hence, the implementations of autonomic control and environmental perception for autonomous driving are outside the scope of this paper. Interested readers are encouraged to refer to our different contributions [2], [3].

In the rest of this section, we present two ICT platforms for the development of cloud-based autonomous driving. The first platform is a smartphone application that controls the vehicle from remote sources. The second platform is also a smartphone application that transfers captured images to the HPC server as fast as possible.

A. Remote Vehicle Control

We have a TOYOTA Prius that is modified to be able to overwrite the control of steering, accelerator, and break from the computer. This computer called "vehicular master computer" is connected to an supplemental embedded device board that can directly send signals to the internale wire system controlling the vehicle. We omit a detailed description of this autonomous driving system, as our primary focus is the quantification of overhead.

Fig. 2 shows a conceptual architecture of our experimental remote vehicle control system. We make use of a commodity smartphone to send the control command to the vehicular master computer equipped within the vehicle through the network. The smartphone application determines the steering angle from the gyro sensor while the accelerator and the break strokes are controlled by the graphics user interface. Thus, we can intuitively use a smartphone as if it were a game controller of the vehicle remotely.

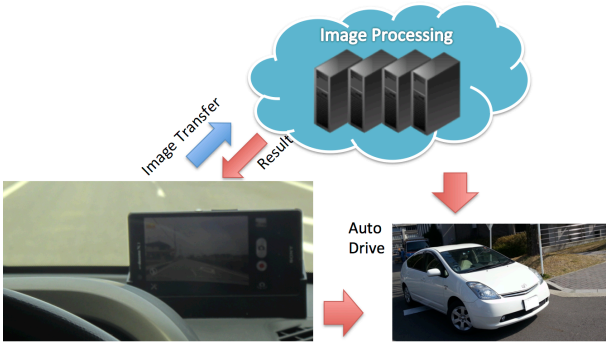

Fig. 3. Networked image processing.

B. Networked Image Processing

Environmental perception is composed of compute-intensive tasks. Perception algorithms are often complex and the volume of input data from laser sensors and high-resolution cameras is not afforable for mobile embedded systems. Therefore it is significant if we can offload these compute-intensive tasks to the cloud in real-time.

Fig. 3 shows a conceptual architecture of our experimental networked image processing system. We capture images in real-time from commodity smartphones attached in the vehicle and transfer them over the network to the HPC server where the actual image processing tasks are executing. The results of image processing such as the detected vehicles and pedestrians are fed back to the vehicular embedded system. Although we have implemented several variants of state-of-the-art image processing algorithms [2], we exclude them from the scope of this paper. Alternatively we investigate if commodity ICT platforms can meet the requirement of throughput and latency for image transfers.

Unlike control commands, the size of an image is large and it generates additional latency for the transfer. For example, each image transfer is composed of (i) capturing an image, (ii) encoding that image, and (iii) transferring the encoded image. Since these three stages can be pipelined, the total throughput would benefit from multithreading and a multicore processor. Furthermore, we have observed that the encoding stage is much longer than the capture and the transfer stages when we use a smartphone as a client due to its wimpy embedded processor performance. This means that the effective rate of image capture and transfer may be limitted to the execution time of encodling. In order to maximize the image transfer throughput on an embedded multicore processor, we increase the number of threads for encoding as shown in Fig. 4. In this example, one can see that the second (blue) transfer does not have to stall before the encoding stage because another thread is available for encoding whereas it would stall due to the preceding encoding process if there is only one thread available for encoding. Thus, we suggest that networked image processing using smartphones should make use of the multithreading capability and the parallelism of embedded multicore processors.

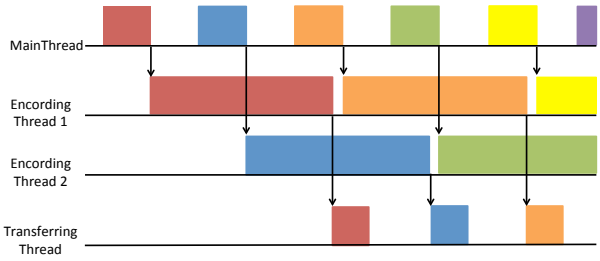

Fig. 4. Multithreading of image transfers.

IV. Evaluation

We now evaluate the data transfer overhead in the context of cloud-based autonomous driving. We assume that autonomic control and environmental perception are provided in the remote server, but they are not within the scope of this paper. What we focus on in this experiment is the measurement of the data transfer overhead. Throughout the experiment, we assume that WiFi (IEEE802.11n 2.4GHz/5.0GHz) is provided with bandwidth of 300Mbps while LTE (au 2.1GHz LTE) achieves 75Mbps for downstream and 25Mbps for upstream.

A. Control Command Transfer

This experiment measures the transfer time taken to send control commands to the vehicle from a smartphone. The commands control the steering, accelerator, and brake of the vehicle. The steering angle is determined by the gyro sensor data while the accelerator and break strokes are manipulated by the graphics user interface of the Andrive application. We use an HTC J butterfly Snapdragon S4 Pro (APQ8064@1.5GHz, Quad Core) smartphone for a testing terminal.

Fig. 5 shows the period (interarrival time) of data transfers for vehicle control commands achieved using WiFi when the vehicular master computer in the vehicle and a client smartphone are synchronized. The nodes are connected within a local area network, and the vehicular master computer sends back an acknowledgment message to the smartphone every time the commands are received for synchronization. The results after a few seconds from an establishing connection between the smartphone and the server are not included since the network is unstable after the establishing connection. It meets a period of $2ms$ on average, which is acceptable for the feedback control rate of autonomous driving [3]. However there are unpredictable spikes that increase the period up to $7ms$ in the worst case. These unpredictable spikes are not acceptable under real-time constraints.

Fig. 6 shows the period of data transfers in the same setup as Fig. 5 except that the vehicular master computer and the smartphone are not synchronized, *i.e.*, the smartphone is sending data without waiting for the acknowledgment message from the vehicular master computer. It is clear that some spikes are removed as compared to Fig. 5.

Figs. 7 and 8 show the period of data transfers for the same set of vehicular control commands using LTE. While WiFi is often suitable for intranet usage, LTE is more publicly

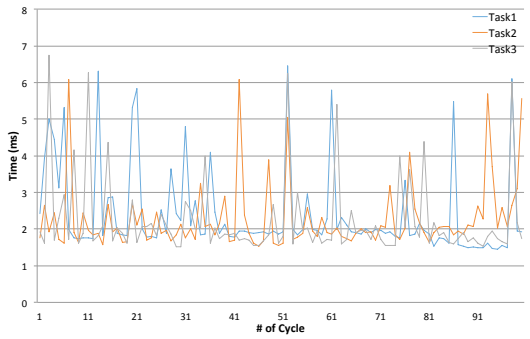

Fig. 5. The achieved period of *synchronous* data transfers for control commands using WiFi.

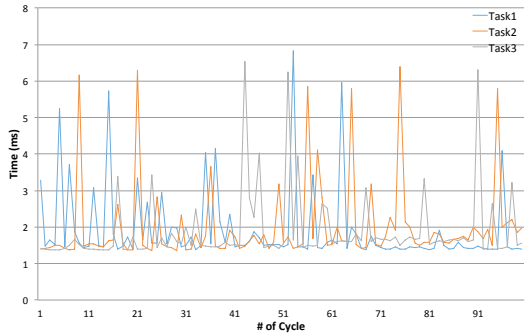

Fig. 6. The achieved period of *asynchronous* data transfers for control commands using WiFi.

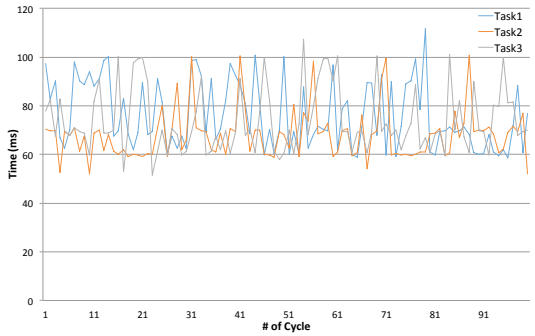

Fig. 7. The achieved period of *synchronous* data transfers for control commands using LTE.

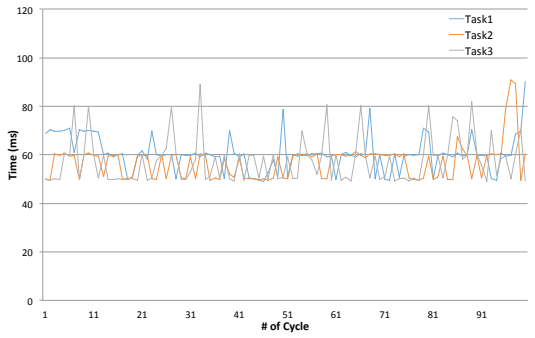

Fig. 8. The achieved period of *asynchronous* data transfers for control commands using LTE.

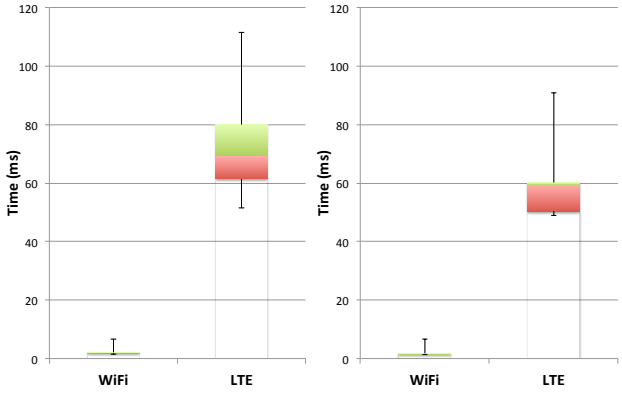

Fig. 9. Summarized box plotting of the periods for *synchronous* (left) and *asynchronous* (right) transfers of control commands.

deployed in the city as part of wireless internet services. As compared to WiFi, the data transfer times achieved by LTE are increased due to its less bandwidth. It takes around $100ms$ to transfer a single set of control commands, and some spike reaches $110ms$ in the worst case when the smartphone waits for an acknowledgment message from the server for every transfer. An asynchronous transfer can mitigate this spike but the average period of data transfers is still capped at around $100ms$. In the evaluation environment of LTE, the difference between synchronous and asynchronous transfers is only around $10ms$ since downstream (the server to the smartphone) is three times faster than upstream (the smartphone to the sever).

Fig. 9 depicts summarized box plotting of the achieved data transfer periods for vehicular control commands. This explains that WiFi is currently a better choice for real-time communication using commodity ICT platforms. A significant difference in performance between WiFi and LTE comes from the fact that LTE is a public service that causes a lot of conflicts with third people while WiFi used in this setup is a private service that is dedicated to our experiment of image transfers. Given this public behavior of LTE, $100ms$ is a reasonable number of the average period of data transfers for control commands. We hope that LTE will become more practical for real-time usage as a publicly available network.

B. Image Transfer

Fig. 10 shows the average frame rate of image transfers achieved using WiFi. We provide six variants of the image transfer program running on the smartphone. As described in Section III, the throughput of image transfers can be improved by multithreading. The top label (Single Thread) uses only a single thread to capture, encode, and transfer images. The other labels represent such implementations that use pipelining with multithreading. Comparing “2Threads (main and transferring)” and “2Threads (main and encoding)“, the result shows that the transferring thread is not efficent when the number of encoding threads is small. Moreover it is speculated that the execution time of the transferring thread is short and

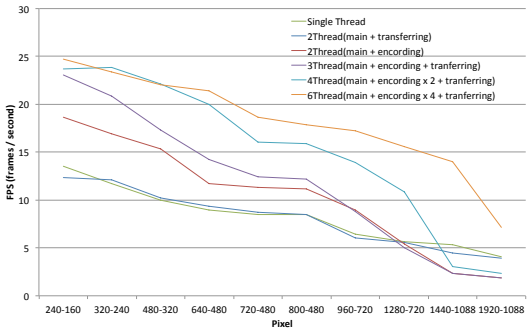

Fig. 10. The frame rate of image transfers using WiFi.

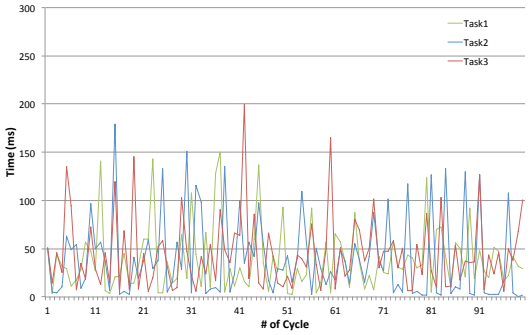

Fig. 11. The achieved period of image transfers using WiFi.

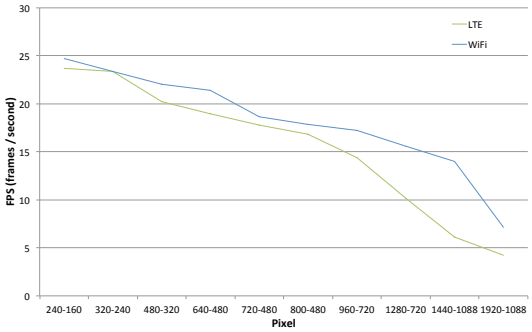

Fig. 12. The frame rates of image transfers using WiFi and LTE respectively.

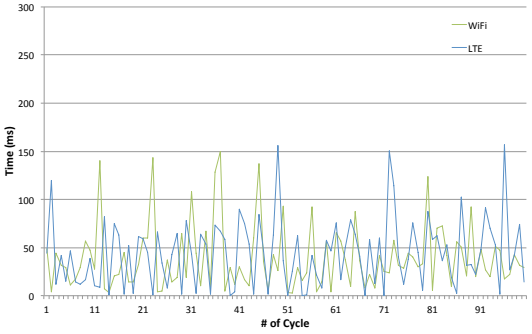

Fig. 13. The periods of image transfers using WiFi and LTE respectively.

finishs during the main thread is waiting for camera capture. Especially the stage of encoding is time consuming, we also use multiple threads for encoding itself. In case of more than image size of 1440×1088 pixels, the single thread is faster than three and four threads. This performance degradation of three and four threads is due to a shortage of memory of an android application when encoding is much slower than the periods of camera capture. However since more than the image size is unusual for smartphone, the problem is not critical in the application. It is notable that using four or six threads in total, we can meet a frame rate of more than 20fps for a standard image size of 640×480 pixels. This is a useful finding that we could use smartphones with WiFi to offload image processing to the cloud in terms of frame rate.

Henceforth we focus on the best performer of the image transfer program, *i.e.*, one represented by a label of "6Threads". Fig. 11 shows the period of image transfers on a cycle basis achieved using WiFi. Albeit high frame rates demonstrated in Fig. 10, the period is actually fluctuating to an extent. For example, some spike reaches $200ms$. This unpredictable spike may jeopardize fine-grained control that actuates based on the results of image processing but it also depends on the algorithm. From this experiment, therefore, we conclude that the algorithm of control using the results of networked image processing must be designed to be dependable against timing jitters.

Fig. 12 shows the frame rate of image transfers achieved using LTE for the same image set as those shown in Fig. 10. For a reference, the frame rate achieved using WiFi is also plotted. We omit the variants of the image transfer program other than "6Threads", since the observed performance trend is similar to that shown in Fig. 10. Unlike the previous experiment of control commands highlighting latency, however, the frame rate of image transfers are not very different between WiFi and LTE. Since we use a smartphone employing an embedded processor, we consider that throughput is more dominated by the processor performance than the network performance. This means that LTE is competitive to WiFi in a scenario that we use a rich smartphone to transfer images to the cloud.

Comparing Figs. 9 and 12, in case of LTE, the results indicates the period of a command (16 bytes) and a small image such as 240×160 or 320×240 pixels are almost the same. This means that it takes time to start tranferring data in the LTE enviroment.

Fig. 13 shows a comparison of the periods of image transfers on a cycle basis for WiFi and LTE. Fig. 14 also shows summarized box plotting of these periods. One can see that the achieved periods of image transfers for WiFi and LTE exhibit similar characteristics. Putting together with the result of frame rate shown in Fig. 12, we conclude that WiFi and LTE are competitive to each other in terms of throughput for image transfers.

Fig. 15 shows performance differences among popular Android devices including Xperia ray (MSM8255@1Ghz, Single Core), REGZA Tablet (Tegra 3@1.3GHz, Quad Core), and HTC J butterfly Snapdragon S4 Pro (APQ8064@1.5GHz, Quad Core), when transferring an image of 640×480 pixels.

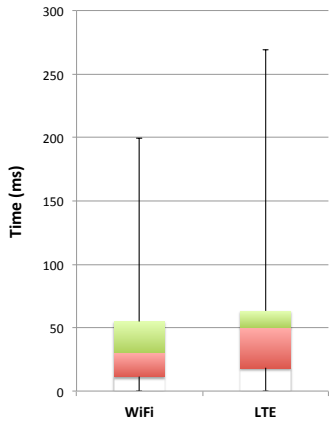

Fig. 14. Summarized box plotting of the periods for image transfers.

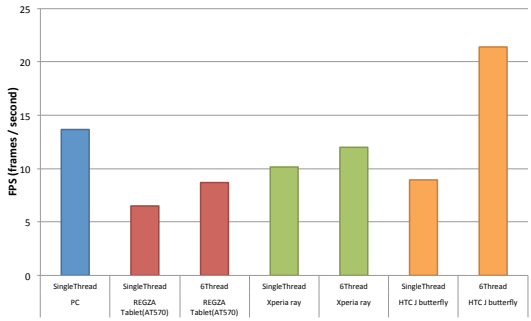

Fig. 15. Performance differences among popular Android devices.

For a reference, we also plot the performance of a single thread performing on a laptop PC with an Intel Core i3 processor (3110M@2.4GHz, Dual Core). Notably non-trivial performance differences are observed in this experiment. Although the class of an employed processor is a primary factor of performance, we experience that the performance of an equipped camera also dominates the overall performance of image transfers. For example, the average number of images per second that the REGZA Tablet can capture is no more than 11.41, while the Xperia ray and the HTC J butterfly can capture 23.20 and 25.74 images per second respectively on average. Therefore the REGZA Tablet does not exhibit high frame rates for entire image transfers, even though it employs a rich processor. Comparing the the Xperia ray and the HTC J butterfly also highlights that a multicore processor benefits from multithreading more significantly. It can be clearly seen that the multithreaded HTC J butterfly even outperforms the single threaded laptop PC.

V. Conclusion

This paper has presented a prototyping of commodity ICT platforms using smartphones and PC servers for a cloud-based autonomous vehicle, which is a representative of mobility CPS applications in the current state of the art. We demonstrated that commodity WiFi could provide a feedback-control period of $5ms$ to $10ms$ for remote autonomic control while a frame rate of 10fps to 20fps for networked image processing. Using commodity LTE, we observed some performance loss from WiFi particularly in latency but it may become useful in a decade as technology advances read today. To the best of our knowledge, this is the first quantitative evidence that explains the availability of commodity ICT platforms for mobility CPS applications.

The tools presented in this paper are all open-source, and may be downloaded from https://github.com/cs005/.

In future work, we plan to prototype a complete cloud-based autonomous vehicle and investigate the availability of commodity ICT platforms using a real case study. We also seek for offloading path planning and/or mission planning tasks to the cloud, given that they could also be computationally expensive as reported in [6]. Further grand challenges include a generalization of "mobility CPS in the cloud" beyond a particular case study of autonomous driving. It is a frontier of mobility CPS to answer the question of what tasks should remain in the mobility agent itself. Since recent powerful compute devices are becoming more suitable for low-latency real-time computing [4], it is important to provide a design nob of hierarchical architectures from the on-board mobility agent through the cloud environment.

References

[1] E. Guizzo11. How Google's Self-Driving Car Works. IEEE Spectrum, 2011.

[2] M. Hirabayashi, S. Kato, M. Edahiro, K. Takeda, T. Kawano, and S. Mita. GPU Implementations of Object Detection using HOG Features and Deformable Models. In *Proc. of the IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*, 2013 (under review).

[3] S. Kagami, S. Thompson, I. Samejima, N. Hatao, Y. Hihei, T. Egawa, T. Hamada, S. Kato, M. Kabasawa, H. Takemura, and H. Mizoguchi. Autonomous Vehicle Navigation by Building 3D Map and by Detecting Humand Trajectory using LIDAR. In *Proc. of the IEEE International Conference on Cyber-Physical Systems, Networks, and Applications*, 2013 (under review).

[4] S. Kato, J. Aumiller, and S. Brandt. Zero-Copy I/O Processing for Low-Latency GPU Computing. In *Proc. of the IEEE/ACM International Conference on Cyber-Physical Systems*, pages 170–178, 2013.

[5] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J.Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun. Towards Fully Autonomous Driving: Systems and Algorithms. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 163–168, 2011.

[6] M. McNaughton, C. Urmson, J. Dolan, and J-W. Lee. Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice. In *Proc. of the IEE International Conference on Robotics and Automation*, pages 4889–4895, 2011.

[7] H. Niknejad, T. Kawano, M. Simizu, and S. Mita. Vehicle detection using discriminatively trained part templates with variable size. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 766–771, 2000.

[8] C. Urmson, J. Anhalt, H. Bae, D. Bagnell, C. Baker, R. Bittner, T. Brown, M. Clark, M. Darms, D. Demitrish, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, S. Kolski, M. Likhachev, B. Litkouhi, A. Kelly, M. McNaughton, N. Miller, J. Nickolaou, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, V. Sadekar, B. Salesky, Y-W. Seo, S. Singh, J. Snider, J. Struble, A. Stentz, M. Taylor, W. Whittaker, Z. Wolkowicki, W. Zhang, and J. Ziglar. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.