International Journal on Advanced Science Engineering Information Technology

Vol.13 (2023) No. 3
ISSN: 2088-5334

Real-Time Bus Arrival Time Estimation API using WebSocket in Microservices Architecture

Kristian Adi Nugraha [a,*]

[a] *Faculty of Information Technology, Universitas Kristen Duta Wacana, Yogyakarta, 55224, Indonesia*
*Corresponding author: *adinugraha@ti.ukdw.ac.id*

***Abstract*—In almost all countries, public transportation is the primary choice for people to make daily trips. One of the main problems of public transportation is the accuracy of the vehicle arrival schedule information. In some types of public transportation with particular tracks, such as rapid transit (e.g., MRT), the arrival time can be predicted easily because traffic jams do not affect transportation. This is different from buses, which use the same route as other vehicles so that traffic jams could affect them. This study tries to solve this problem by building an API that can be used to get information on the estimated arrival time of the bus. Estimated time is calculated based on the condition of the bus position and the actual condition of highway traffic from Google Directions API. The API was built using microservices technology, so it can be done quickly if it is further developed to a larger scale. The test was conducted on the Trans Jogja Bus by taking two routes, the short and the long routes. Each route could be explored 20 times under varying time and traffic conditions. Then, the difference between the estimated bus arrival time and the actual bus arrival time could be calculated on each trip. Based on the test results, the estimated bus arrival time generated by the API can be said to be entirely accurate because the difference between the estimated time and the actual bus arrival time is less than 30 seconds.**

***Keywords*— API; microservices; software architecture; bus arrival time; WebSocket.**

Manuscript received 22 Jul. 2022; revised 7 Nov. 2022; accepted 5 Jan. 2023. Date of publication 30 Jun. 2023.

I. INTRODUCTION

Information is essential and must be obtained quickly through advanced technologies, primarily smartphone-based applications because three out of eight billion people today already have smartphones [1]. Any information related to public transportation systems, such as routes and schedules, is an example of an essential piece of information. If someone has the necessary transportation details, it could help them to avoid being late. The most needed information related to public transportation systems is the arrival time of vehicles at the station so that people can figure out how long it could take them to get to their destination [2]. Commuters use two kinds of mass transportation in the city: bus and rapid transit (MRT, subway). Most regions, particularly Southeast Asia, only have buses as the primary means of traveling [3]. A problem appeared when discussing the arrival time of the bus because it shared the road with other vehicles and did not have any particular track as rapid transit did. As a result, it is difficult for any bus to predict its arrival time because it depends on many factors like traffic jams, traffic diversion, or roadworks [4]. That seems impossible to make a fixed schedule for the bus arrival time because the time would be different each time the bus arrived. The only way to fix this problem is by periodically measuring bus arrival time estimation to the bus stop based on road traffic or other obstacles that might slow down the bus speed. Although this method might not accurately predict the arrival time, at least the estimated time result can help compared to none.

The solution above requires a GPS tracker inside the bus with a communication device to periodically send its location to the server to estimate the bus arrival time[5]. However, this approach is costly and challenging to implement because needs to buy a set of hardware for each bus. A cheaper alternative to applying that approach is a smartphone-based application to send its location through cellular networks to the server. There is no need to buy additional equipment to implement this approach, assuming each driver already owns a mobile phone (smartphone). This approach also requires a web-based server API (Application Programming Interface), which connects the client and server to store bus location data from the driver's smartphone and calculate the arrival time to the next bus stop [6].

Service-oriented architecture is a type of architecture that is widely used to accommodate applications that have two sides, called client and server [7]. There are two popular types of server-based architecture: monolithic and microservices. Monolithic architecture is a traditional server-based architecture that can be quickly designed and implemented since all components are placed in one (mono) computer [8]. Although it has many benefits, monolithic architecture has some drawbacks, such as scaling up and maintaining, that are hard to deal with. We sometimes did not know how many resources the server needed initially, so it would be a waste if we set up huge but unused resources. Microservices architecture was designed to resolve this issue. It consists of many modules that can communicate to exchange data, and each module stands for only one loosely coupled service [9]. If a module was processing too many tasks, that module's resources could be upgraded to handle the job properly without disturbing another module.

Because bus arrival schedule information must be presented in real-time, a particular technology, called WebSocket, is needed to support this solution. WebSocket is a two-way communication technology that allows the client or server to send messages first through a single TCP/IP socket connection [10]. Thus, anyone who wants to send information can send it directly without waiting for the other side to call first. Because the data transmission process can be done quickly, this technology is suitable for application to a system that requires real-time information. This research aims to see how accurate the estimated time of bus arrival is from a socket-based server in a microservice architecture; then, it can be used to help people by providing real-time bus schedule information.

II. Material and Method

In this section, socket server and microservices architecture are presented.

A. Application Programming Interface (API)

Application Programming Interface or API is an interface that allows a computer to communicate with other computers to exchange data [11, 12, 13]. API is usually used for software built with a service-oriented architecture structure consisting of a client and a server, where the API acts as a server [14]. All communication between machines is done through the API, but currently, the word "API" refers more to a web-based API [15, 16, 17]. Web APIs, often called Web Service [18], are usually used as the back-end to build applications with various platforms, such as web-based and mobile applications because the API is designed not to be tied to the platform type of client that could access it [19, 20]. One of the most widely used API types is REST because of its simple scheme [21, 22, 23]. It does not require powerful resources, so server performance is not too heavy [24]. Only one centralized database is placed on the web server using the API, as shown in Figure 1.

While from the client side (web and mobile), it is only enough to make a request when it needs to send data and receive a response when it requires specific data from the database. The entire communication process is generally done over the Internet using the HTTP protocol.

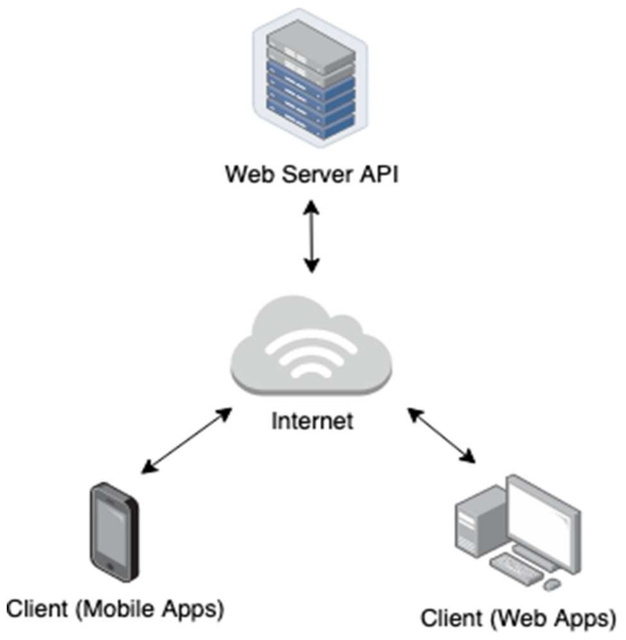


Fig. 1 Service-oriented architecture diagram

B. WebSocket

Hypertext Transfer Protocol (HTTP) is the most popular network protocol to distribute information from one machine to another over a network [25].

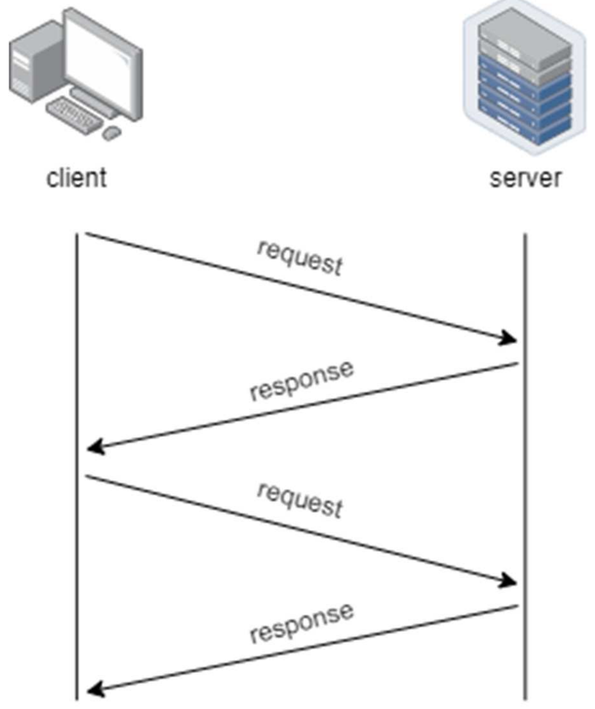


Fig. 2 Standard HTTP protocol

WebSocket is a network protocol like HTTP. It is part of a computer program that allows full-duplex communication between two programs through the network, usually implemented in web or mobile applications to communicate with the web [26, 27]. WebSocket consists of an IP address and port as an endpoint to listen to any requests sent by the client. The main distinction between HTTP and WebSocket protocols is that each request is always replied to with a single response (handshake), as shown in Figure 2. In contrast, the WebSocket protocol can reply to each request with many responses. The server must receive the request first before responding to the client using the HTTP protocol.

On the other hand, the WebSocket protocol allows the server to transmit responses at any moment, even if the client does not initiate a request (bidirectional) [28]. Because the data transmission process is faster than the standard HTTP protocol, WebSocket can be used to build a system that requires real-time information transmission [29]. In this study, WebSocket could continuously be used to update information on the location of the bus that is running and could also be used to obtain real-time bus arrival schedule information at a bus stop.

C. *Microservice*

Microservice architecture is a collection of software components composed of small modules [30]. Microservices have the advantage of adapting to the system's scale, so it does not matter if the system scale suddenly needs to be expanded or reduced [31, 32, 33]. Microservice can divide the work into small modules, so the load on each module could be lighter [34, 35]. The main advantage of using microservices is the ease of maintenance and flexibility when system changes need to be implemented [36, 37].

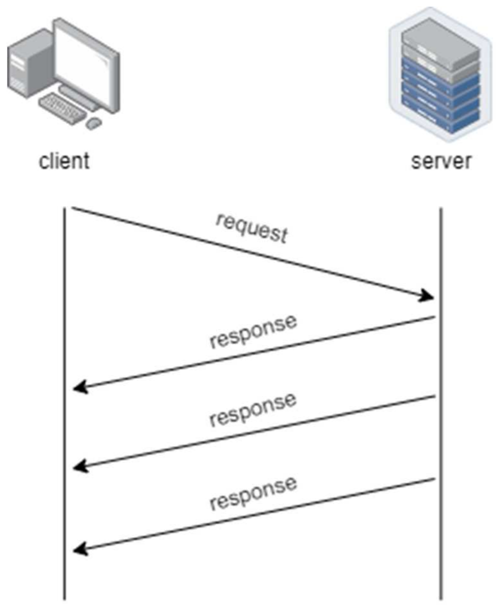


Fig. 3 WebSocket protocol

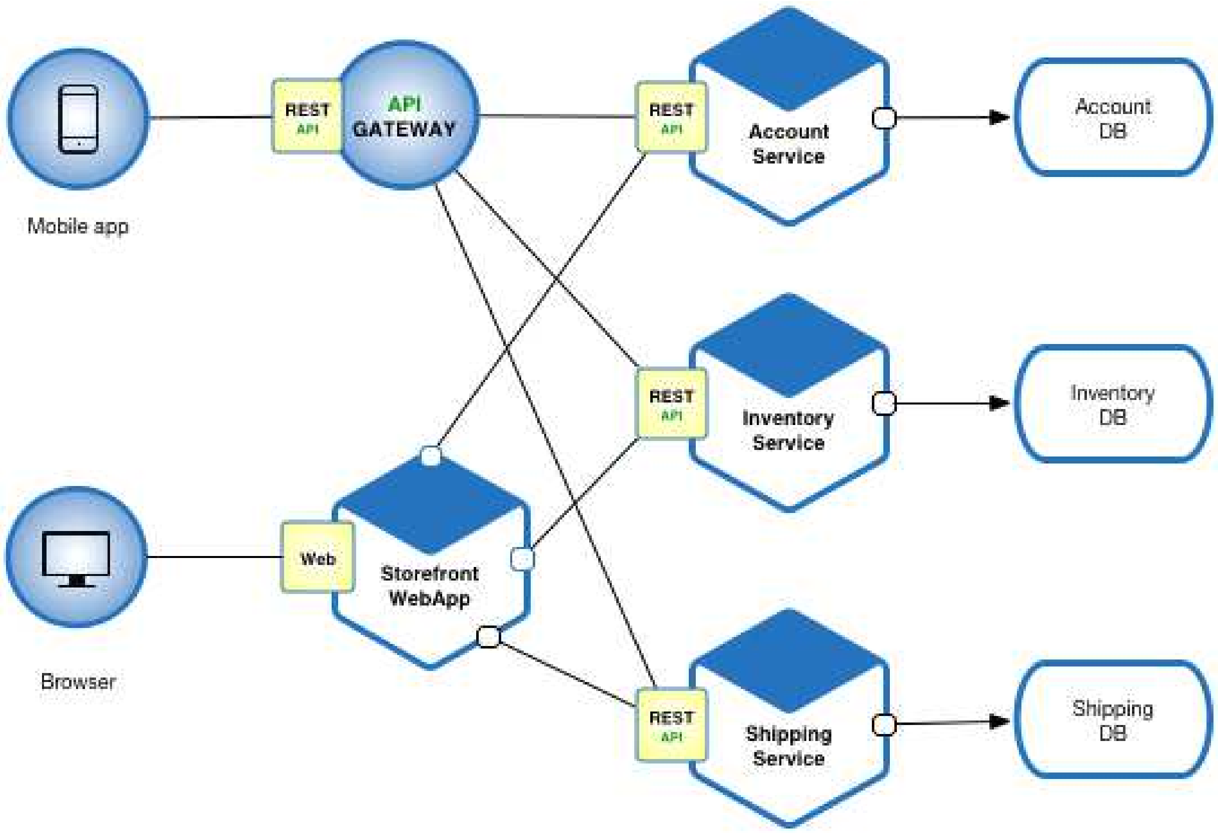


Fig. 4 Example of microservice architecture [38]

D. *System Preparation*

The system architecture consists of several servers to process requests and one API Gateway as the application's endpoint. All client requests could be addressed to the API Gateway; then, the API Gateway could determine the server that could process the request. Thus, the server workload can be divided evenly and not burden only one server, as shown in Figure 5. Because it uses a microservices architecture, if the server load is considered too heavy, for example, because the number of bus fleets or end-users increases, it is enough to add a new server connected to the existing system. This architecture also allows the API to be used for similar transportation systems in other cities by copying existing services to a new server and connecting them all into a system.

There are two types of clients who could use the API: bus drivers and end-users. The bus driver's application (client) updates the bus's latitude and longitude (geolocation) coordinate continuously with time intervals of 10 seconds through the background (asynchronous) process. The location could be sent and stored on the server via the location service bus using the socket protocol. The 10-second break was chosen with consideration so as not to burden the driver's cellphone too much, especially in terms of power and bandwidth. In addition, if the bus travels at a standard speed of 40 kilometers per hour, then in 10 seconds, the bus can move the maximum distance of 111 meters. This value could be the maximum value of the location difference between the latest bus location data and the actual bus location still within the tolerance limit.

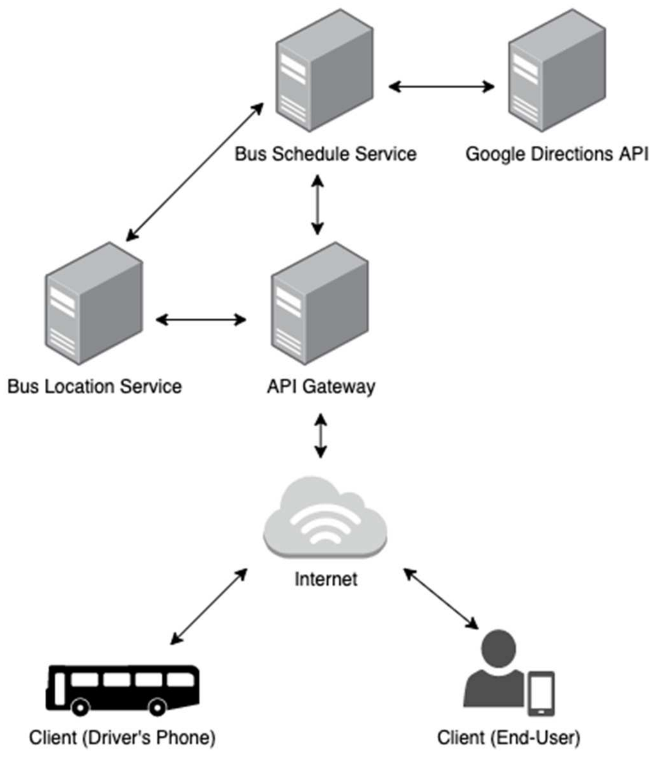
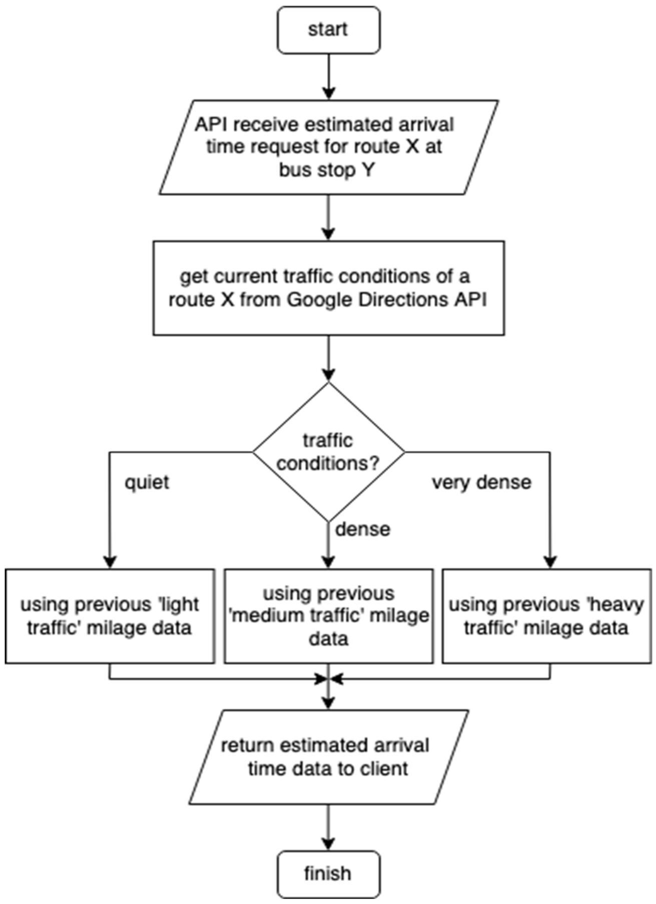
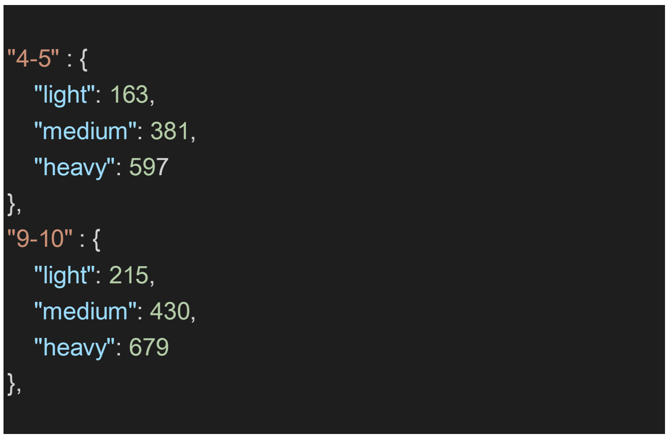


Fig. 5 System architecture

In the end-user application, there is a feature to view the estimated time of bus arrival at a bus stop via bus schedule service. As a client, the end-user application could send a request to the server (API) to get the estimated arrival time data, as shown in Figure 6 Estimated arrival time is calculated based on previous mileage data compared to current traffic conditions obtained from the Google Directions API. There are three traffic density categories: light, medium, and heavy. All three are stored separately on the database in JSON format, as in the example in Figure 7. JSON can produce smaller file sizes than other formats (e.g., XML), so it's less burdensome on the server [39].

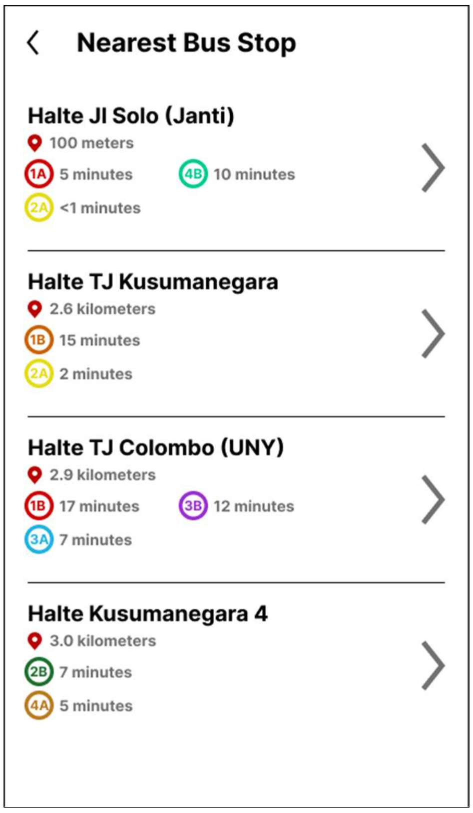

start
API receive estimated arrival time request for route X at bus stop Y
get current traffic conditions of a route X from Google Directions API
traffic conditions?
quiet
dense
very dense
using previous 'light traffic' milage data
using previous 'medium traffic' milage data
using previous 'heavy traffic' milage data
return estimated arrival time data to client
finish
Fig. 6 Flowchart of estimated arrival time request



```
"4-5" : {
    "light": 163,
    "medium": 381,
    "heavy": 597
},
"9-10" : {
    "light": 215,
    "medium": 430,
    "heavy": 679
},
```

Fig. 7 Example of traffic data

Each data is stored in seconds with a key in the form of the origin and destination stop id, connected with a '-' sign. If the current traffic conditions are heavy, then the data that could be used for estimation is when the traffic conditions are heavy. Thus, the estimated bus arrival time calculation is expected to produce a more accurate value.


Nearest Bus Stop

Halte Jl Solo (Janti)
100 meters
1A 5 minutes
4B 10 minutes
2A <1 minutes

Halte TJ Kusumanegara
2.6 kilometers
1B 15 minutes
2A 2 minutes

Halte TJ Colombo (UNY)
2.9 kilometers
1B 17 minutes
3B 12 minutes
3A 7 minutes

Halte Kusumanegara 4
3.0 kilometers
2B 7 minutes
4A 5 minutes

Fig. 8 Example of API usage for end-user apps

Every time the bus moves from one stop to another, the server could save the current travel time and traffic conditions to update the data for calculating the next estimated arrival time. Considering the estimated arrival time of the bus could be more accurate because the road conditions are not much different from the previous trip. An example of an application user interface that could use this service is shown in Figure 8.

III. Result and Discussion

The test was conducted using the Trans Jogja Bus public transportation service by riding the bus directly. There are two routes that are the object of the test: a short route that passes through 8 stops and a long route that passes through 15 stops. Each route could take 20 trips on different days with varying times and traffic conditions.

TABLE I
ESTIMATED ARRIVAL TIME DIFFERENCE FOR SHORT ROUTE

# Bus Stop	Distance (m)	Time Difference (in seconds)		
		Min	Max	Average
1	700	1	40	19.7
2	1500	1	42	14.6
3	2100	3	82	34.7
4	1300	-9	47	22.7
5	950	-15	58	27.35
6	1200	1	59	23.95
7	1000	-7	47	27.25
8	1200	-11	57	24.7
Average		-4.5	54	24.37

Every time the bus departs from one stop to the next, the estimated time shown by the application could be compared with the actual time to the next stop. Thus, each stop could have as many as 20 estimated times varying from light to heavy traffic. The results of these tests are shown in Table 1 for the short route and Table II for long route.

From Table 1 above, we can conclude that the average overall difference between the estimated bus arrival time and the actual time is under 30 seconds, to be exact, 24.37 seconds. There is only one bus stop with an average time difference of more than 30 seconds, namely at bus stop #3, with a value of 34.7 seconds. At that stop, the highest time difference is 82 seconds, where this value has a fairly large gap when compared to other stops with a shorter distance. It can happen because the distance between bus stop #3 and the previous stop (#2) is quite far compared to the distance between other stops, which is 2100 meters (2.1 km). Because the journey taken is getting farther, the possibility of obstacles could also be greater, which delays the bus arrival time.

TABLE II
ESTIMATED ARRIVAL TIME DIFFERENCE FOR LONG ROUTE

# Bus Stop	Distance (m)	Time Difference (in seconds)		
		Min	Max	Average
1	900	4	58	24.25
2	800	6	59	32.4
3	1100	2	58	29.0
4	1000	-24	28	11.2
5	450	-1	20	9.35
6	1350	9	89	51.95
7	500	3	55	28.7
8	550	1	54	23.6
9	500	-5	53	25.35
10	900	1	54	23.4
11	900	8	54	30.15
12	1100	2	58	20.25
13	750	1	36	15.15
14	700	-18	56	23.9
15	950	5	45	18.3
Average		-0.4	51.8	24.46

Similar results also occur in the test for long routes, where the average difference between the estimated arrival time and the actual time is 24.46 seconds. There are several stops with an estimated bus arrival time of more than 30 seconds, but the most significant is at stop #6, where the average time difference is 51.95 seconds. Stop #6 has the longest distance to the previous stop compared to other stops, resulting in a significant time difference.

The relationship between the distance to the difference between the estimated arrival time and the actual time can be seen as a plot in Figure 9. The figure shows that the farther the distance between the stops, the higher the difference in time difference could be, although the increase is not significant. It happens because the farther the bus travels from one stop to the next, the higher the chance of problems hindering bus travel. On the other hand, if the bus mileage is relatively short, the bus could reach the next stop faster, so the chance of obstacles appearing is also relatively small.

Based on the overall test results, the average difference between the estimated arrival time and the actual time is under 30 seconds. Because the error value is relatively small, the results can be tolerated so that the estimated bus arrival time can still be said to be accurate enough to be widely used.

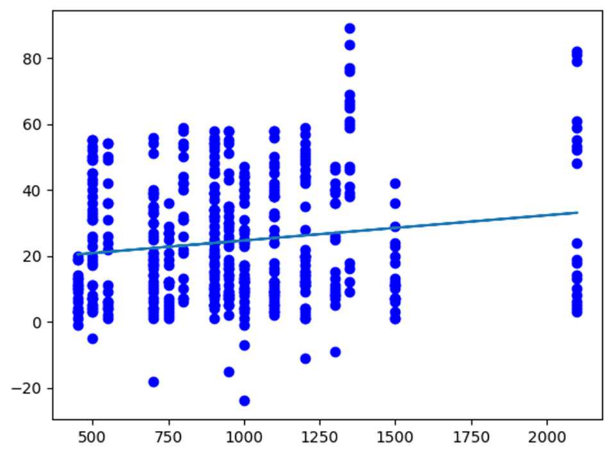


Fig. 9 Relations between distance (x) with difference between estimated and actual arrival time (y)

IV. Conclusion

Overall, it can be concluded that the estimated bus arrival time information can be given accurately with an error rate of under 30 seconds. If there is a difference between the estimated time and the actual arrival time, then the difference could not be more than 30 seconds, which can be said to be a good result. Some steps that can be taken in the future to improve accuracy are by enriching traffic information from sources other than the Google Directions API. Since road traffic is unpredictable, it would be better if there were many other sources of information to determine the estimated bus arrival time in real-time.

Acknowledgment

We thank Universitas Kristen Duta Wacana for providing the sponsorship to do this research.

References

[1] S. O'Dea, "Smartphone users worldwide 2016-2021," Statista, 10 December 2010. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/#:~:text=How%20many%20people%20have%20smartphones,in%20the%20next%20few%20years.. [Accessed 12 February 2021].

[2] B. Yu, W. H. K. Lam and M. L. Tam, "Bus arrival time prediction at bus stop with multiple routes," *Transportation Research Part C: Emerging Technologies,* vol. 19, no. 6, pp. 1157-1170, 2011.

[3] R. Hassan, H. I. A. Jabar, M. K. Hasan, M. C. Lam and W. M. H. W. Hussain, "Cloud Based Performance Data Analysis and Monitoring System for Express Bus in Malaysia," *International Journal on Advanced Science, Engineering and Information Technology,* vol. 9, no. 6, pp. 1959-1967, 2019.

[4] Z. Y. Xie, Y. R. He, C. C. Chen, Q. Q. Li and C. C. Wu, "Multistep Prediction of Bus Arrival Time with the Recurrent Neural Network," *Mathematical Problems of Applied System Innovations for IoT Applications,* vol. 2021, 2021.

[5] Q. Han, K. Liu, L. Zeng, G. He, L. Ye and F. Li, "A Bus Arrival Time Prediction Method Based on Position Calibration and LSTM," *IEEE Access,* vol. 8, pp. 42372 - 42383, 2020.

[6] D. Sebastian, Restyandito and K. A. Nugraha, "Developing of Middleware and Cross Platform Chat Application," *International Journal of Advanced Computer Science and Applications,* vol. 12, no. 11, pp. 79-85, 2021.

[7] V. A. Wardhany, H. Yuliandoko, Subono, M. U. H. A and I. G. P. Astawa, "Smart System and Monitoring of Vanammei Shrimp Ponds," *International Journal on Advanced Science, Engineering and Information Technology,* vol. 11, no. 4, pp. 1366-1372, 2021.

[8] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou and X. Shen, "Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach," *IEEE Transactions on Mobile Computing,* vol. 20, no. 3, pp. 939 - 951, 2021.

[9] Y. Y. F. Panduman, M. R. U. Albaab, A. R. A. Besari, S. Sukaridhoto, A. Tjahjono and R. P. N. Budiarti, "Implementation of Data Abstraction Layer Using Kafka on SEMAR Platform for Air Quality Monitoring," *International Journal on Advanced Science, Engineering and Information Technology,* vol. 9, no. 5, pp. 1520-1527, 2019.

[10] M. G. Son, K. S. Park and Y. H. Kong, "An Improvement of Hospital Reception System using Web Socket," *Journal of the Korea Society of Computer and Information,* vol. 20, no. 1, pp. 185-195, 2015.

[11] J. Zhang, H. Jiang, Z. Ren, T. Zhang and Z. Huang, "Enriching API Documentation with Code Samples and Usage Scenarios from Crowd Knowledge," *IEEE Transactions on Software Engineering,* vol. 47, no. 6, pp. 1299 - 1314, 2019.

[12] Y. Zhou, C. Wang, X. Yan, T. Chen, S. Panichella and H. Gall, "Automatic Detection and Repair Recommendation of Directive Defects in Java API Documentation," *IEEE Transactions on Software Engineering,* vol. 46, no. 9, pp. 1004 - 1023, 2020.

[13] C. Chen, Z. Xing, Y. Liu and K. O. L. Xiong, "Mining Likely Analogical APIs Across Third-Party Libraries via Large-Scale Unsupervised API Semantics Embedding," *IEEE Transactions on Software Engineering,* vol. 47, no. 3, pp. 432 - 447, 2021.

[14] L. Qi, Q. He, F. Chen, X. Zhang, W. Dou and Q. Ni, "Data-Driven Web APIs Recommendation for Building Web Applications," *IEEE Transactions on Big Data,* vol. 8, no. 3, pp. 685 - 698, 2022.

[15] L. Li, W. Chou, W. Zhou and M. Luo, "Design Patterns and Extensibility of REST API for Networking Applications," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 154 - 167, 2016.

[16] X. Wang, Q. Sun and J. Liang, "JSON-LD Based Web API Semantic Annotation Considering Distributed Knowledge," *IEEE Access* , vol. 8, pp. 197203 - 197221, 2020.

[17] L. Shen, M. Pan, L. Liu, D. You, F. Li and Z. Chen, "Contexts Enhance Accuracy: On Modeling Context Aware Deep Factorization Machine for Web API QoS Prediction," *IEEE Access,* vol. 8, pp. 165551 - 165569, 2020.

[18] J. M. Z. Mohd Hariz Naim, K. A. Jalil and L. Salahuddin, "Segmented Network Architecture for Promoting High Availability in Fog Computing through Middleware," *International Journal on Advanced Science, Engineering and Information Technology,* vol. 11, no. 6, pp. 2509-2517, 2021.

[19] M. Lamothe, W. Shang and T.-H. P. Chen, "A3: Assisting Android API Migrations Using Code Examples," *IEEE Transactions on Software Engineering* , vol. 48, no. 2, pp. 417 - 431, 2020.

[20] W. Rafique, X. Zhao, S. Yu, I. Yaqoob, M. Imran and N. U. N. C. Wanchun Dou Department of Computer Science and Technology and the State Key Laboratory for Novel Software Technology, "An Application Development Framework for Internet-of-Things Service Orchestration," *IEEE Internet of Things Journal,* vol. 7, no. 5, pp. 4543 - 4556, 2020.

[21] A. Neumann, N. Laranjeiro and J. Bernardino, "An Analysis of Public REST Web Service APIs," *IEEE Transactions on Services Computing,* vol. 14, no. 4, pp. 957 - 970, 2021.

[22] G. Vega-Gorgojo, "CRAFTS: Configurable REST APIs for Triple Stores," *IEEE Access* , vol. 10, pp. 32426 - 32441, 2022.

[23] L. Jiang, H. Liu, H. Jiang, L. Zhang and H. Mei, "Heuristic and Neural Network Based Prediction of Project-Specific API Member Access," *IEEE Transactions on Software Engineering,* vol. 48, pp. 1249 - 1267, 2022.

[24] N. Laranjeiro, J. Agnelo and J. Bernardino, "A Black Box Tool for Robustness Testing of REST Services," *IEEE Access,* vol. 9, pp. 24738 - 24754, 2021.

[25] Y. Go, H. Noh, G. Park and H. Song, "Hybrid TCP/UDP-Based Enhanced HTTP Adaptive Streaming System With Multi-Homed Mobile Terminal," *IEEE Transactions on Vehicular Technology,* vol. 68, no. 5, pp. 5114 - 5128, 2019.

[26] Q. Liu and X. Sun, "Research of Web Real-Time Communication Based on Web Socket," *International Journal of Communications, Network and System Sciences,* vol. 5, no. 12, pp. 797-801, 2012.

[27] C. U. o. P. a. T. C. C. Wenbo Mei School of Computer Science and Technology and Z. Long, "Research and Defense of Cross-Site WebSocket Hijacking Vulnerability," in *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, Dalian, China, 2020.

[28] B. C. Li and S. Z. Yu, "Keyword Mining for Private Protocols Tunneled Over WebSocket," *IEEE Communications Letters,* vol. 20, no. 7, pp. 1337 - 1340, 2016.

[29] C. Pintavirooj, T. Keatsamarn and T. Treebupachatsakul, "Multi-Parameter Vital Sign Telemedicine System Using Web Socket for COVID19 Pandemics," *Healthcare,* vol. 9, no. 3, 2021.

[30] H. Calderón-Gómez, L. Mendoza-Pittí, M. Vargas-Lombardo, J. M. Gómez-Pulido and J. Luis, "Telemonitoring System for Infectious Disease Prediction in Elderly People Based on a Novel Microservice Architecture," *IEEE Access,* vol. 8, pp. 118340 - 118354, 2020.

[31] N. C. Coulson, S. Sotiriadis and N. Bessis, "Adaptive Microservice Scaling for Elastic Applications," *IEEE Internet of Things Journal,* vol. 7, no. 5, pp. 4195 - 4202, 2020.

[32] A. Ali and M. M. Iqbal, "A Cost and Energy Efficient Task Scheduling Technique to Offload Microservices Based Applications in Mobile Cloud Computing," *IEEE Access,* vol. 10, pp. 46633 - 46651, 2022.

[33] Y. Liang and Y. Lan, "TCLBM: A task chain-based load balancing algorithm for microservices," *Tsinghua Science and Technology ,* vol. 26, no. 3, pp. 251 - 258, 2021.

[34] K. Fu, W. Zhang, Q. Chen, D. Zeng and M. Guo, "Adaptive Resource Efficient Microservice Deployment in Cloud-Edge Continuum," *IEEE Transactions on Parallel and Distributed Systems,* vol. 33, no. 8, pp. 1825 - 1840, 2022.

[35] W. Lv, Q. Wang, P. Yang, Y. Ding, B. Yi, Z. Wang and C. Lin, "Microservice Deployment in Edge Computing Based on Deep Q Learning," *IEEE Transactions on Parallel and Distributed Systems ,* vol. 33, no. 11, pp. 2968 - 2978, 2022.

[36] I. K. Aksakalli, T. Celik, A. B. Can and B. Tekinerdogan, "Systematic Approach for Generation of Feasible Deployment Alternatives for Microservices Publisher: IEEE Cite This PDF," *IEEE Access ,* vol. 9, pp. 29505 - 29529, 2021.

[37] G. Blinowski, A. Ojdowska and A. Przybyłek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access,* vol. 10, pp. 20357 - 20374, 2022.

[38] C. Richardson, "Pattern: Microservice Architecture," Microservices.io, 2021. [Online]. Available: https://microservices.io/patterns/microservices.html. [Accessed 30 November 2021].

[39] Rianto, M. A. Rifansyah, R. Gunawan, I. Darmawan and A. Rahmatulloh, "Comparison of JSON and XML Data Formats in Document Stored NoSql Database Replication Processes," *International Journal on Advanced Science, Engineering and Information Technology,* vol. 11, no. 3, pp. 1150-1156, 2021.