applied sciences

MDPI

Article

Prefetching Method for Low-Latency Web AR in the WMN Edge Server

Seyun Choi [1], Sukjun Hong [1], Hoijun Kim [2], Seunghyun Lee [3] and Soonchul Kwon [4,*]

[1] Department of Smart System, Kwangwoon University, Seoul 01897, Republic of Korea
[2] Department of Plasma Bio Display, Kwangwoon University, Seoul 01897, Republic of Korea
[3] Department of Ingenium College Liberal Arts, Kwangwoon University, Seoul 01897, Republic of Korea
[4] Graduate School of Smart Convergence, Kwangwoon University, Seoul 01897, Republic of Korea
* Correspondence: ksc0226@kw.ac.kr; Tel.: +82-2-940-8637

Abstract: Recently, low-latency services for large-capacity data have been studied given the development of edge servers and wireless mesh networks. The 3D data provided for augmented reality (AR) services have a larger capacity than general 2D data. In the conventional WebAR method, a variety of data such as HTML, JavaScript, and service data are downloaded when they are first connected. The method employed to fetch all AR data when the client connects for the first time causes initial latency. In this study, we proposed a prefetching method for low-latency AR services. Markov model-based prediction via the partial matching (PPM) algorithm was applied for the proposed method. Prefetched AR data were predicted during AR services. An experiment was conducted at the Nowon Career Center for Youth and Future in Seoul, Republic of Korea from 1 June 2022 to 31 August 2022, and a total of 350 access data points were collected over three months; the prefetching method reduced the average total latency of the client by 81.5% compared to the conventional method.

Keywords: wireless mesh network; edge server; augmented reality; prefetching

check for updates

Citation: Choi, S.; Hong, S.; Kim, H.; Lee, S.; Kwon, S. Prefetching Method for Low-Latency Web AR in the WMN Edge Server. *Appl. Sci.* **2023**, *13*, 133. https://doi.org/10.3390/app13010133

Academic Editors: Radu Comes, Jing-Jing Fang, Dorin-Mircea Popovici and Calin Gheorghe Dan Neamtu

Received: 23 November 2022
Revised: 19 December 2022
Accepted: 20 December 2022
Published: 22 December 2022

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Studies have been conducted on augmented reality (AR) services in web browsers such as the WebXR Device API (Web eXtended Reality Device Application Programming Interface) [1], AR.js [2], and Three.js [3]. Users can access AR services through the web browser on their smartphone without installing a separate application; this helps improve user accessibility [4]. However, there are limitations in that users must always be connected to the services [5], and fetching causes delays when using an AR service [6].

The Institute of Electrical and Electronics Engineers (IEEE) has defined a standard called 802.11s, which is related to the mesh for a wireless mesh network (WMN). 802.11s conducts traffic forwarding via 802.11ac [7–10]. Internet wireless fidelity (Wi-Fi) can be provided without a shadow area throughout the service area that provides the AR service. Web data are saved on an edge server with a short physical distance hop. The client fetches the saved web data, which helps reduce latency. When downloading AR data, the edge server communicates with the client without a wide area network (WAN), and therefore, it is not affected by WAN latency [11,12]. Further, it reduces the latency required to access the server and operates at a stable and constant speed. The AR data can be downloaded relatively quickly [13]; however, if there is a considerable amount of high-capacity AR data, latency can still occur [14].

This study attempted to reduce user latency in AR services. To this end, we proposed a prefetching method that uses the Markov model. The Markov model enables predictions via the partial matching algorithm used for prefetching [15]. The proposed method considers the priority of each AR dataset to reduce latency, and it allows the prediction of the AR data that the user will request next. According to the proposed method, AR data are sequentially downloaded [16]. In this study, the following experiments were conducted: (1) The latency

was measured according to the use of an edge server; (2) a comparative analysis of network traffic was performed; (3) the hit ratio and latency for each request, user, and data in the time order were measured; and (4) the waste ratio results for each user were calculated.

The remainder of this paper is organized as follows: Section 2 explains the background theory. Section 3 describes the proposed methods. Section 4 presents the experimental environment, evaluation method, and results. Finally, Section 5 presents the conclusions, limitations, and future research directions.

2. Background Theory

2.1. WebAR Service

AR allows the seamless integration of virtual data with the real world to provide users with sensory experiences that transcend reality [14,17]. AR is built and provided as a mobile application; AR data are pre-stored inside a client terminal and serviced [18–20]. The WebAR service starts downloading AR data when a webpage is accessed; therefore, this service does not require separate application installations compared to those required by native AR applications. Currently, the representative technologies include WebXR Device API, AR.js, and Three.js [21]. Several studies have focused on WebAR. For example, Qiao [14] conducted a study on a mechanism for the implementation of WebAR on mobile devices and presented various approaches to implement Web AR. Rodrigues [22] conducted a study using WebAR to use several types of media as AR objects.

2.2. Wireless Mesh Network

A WMN is a communication network that comprises a mesh router and mesh client; it provides broadband internet access, wireless local area network (LAN) coverage, and network connectivity to both mobile and stationary nodes. The WMN is the most efficient wireless technology compared to general networks such as ad hoc sensor networks [23]. Further, wireless 802.11 mesh networks have the advantages of low cost, easy and incremental deployment, and fault tolerance [24,25]. A WMN can be reconfigured dynamically. The nodes can automatically establish and maintain mesh connections internally, and it has advantages such as improved stability [26]. Thus far, various studies have been conducted on WMNs. For example, Benyamina [27] conducted a study to improve the performance of a WMN network design, and Akyildiz et al. [28] studied the protocol for a WMN. Figure 1 shows the wireless mesh network topology.

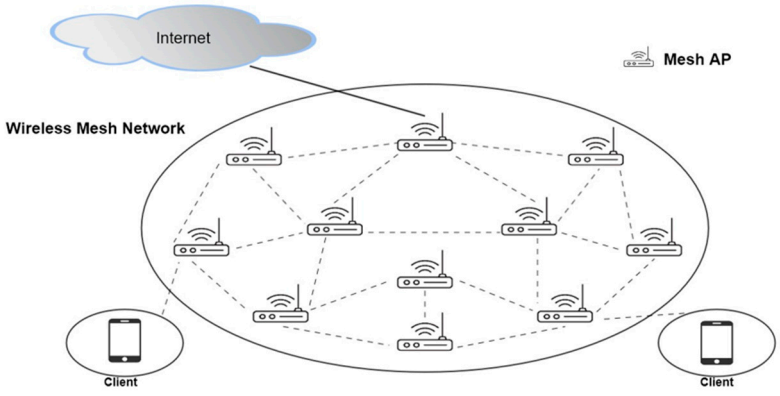


Figure 1. Wireless mesh network topology.

2.3. Edge Server

An edge server is a server on the edges of the network [29]; it is located where the corresponding function is required and distributed processing is performed [30]. The edge server performs compute offloading, data storage, caching, and processing. Further, it distributes request and delivery services from the cloud to the user [31].

Figure 2a shows the network distance between the client and cloud server, and Figure 2b shows the network distance between the client and edge server. The physical distance between the edge server and client is shorter than that between the cloud server and client. The edge server simplifies the network structure, and the time required to send and receive data is short [32]. The edge servers in the LAN layer can communicate data with clients in a stable manner. In the case of a cloud server, the data are transmitted from the client to the destination server via a metropolitan area network (MAN) and WAN. Latency such as bottlenecks and intermediate node systems can occur depending on the amount of data transmission [33]. Sukhman [34] conducted research on 5G, edge caching, and computing. Edge caching has been studied in the existing literature to minimize latency and load.

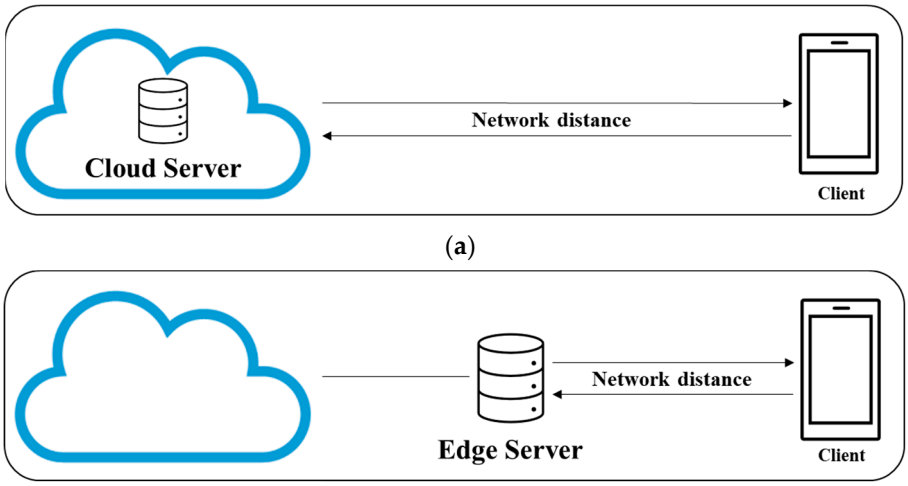


Figure 2. Network type and physical distance: (**a**) cloud server; (**b**) edge server.

2.4. Prefetching

Caching and prefetching methods have been proposed for the prediction of data usage patterns and to fetch data from a location close to the user in advance. The caching method sends cached data through a proxy server; this saves frequently requested data closer to the user, and it aims to reduce bandwidth consumption, network congestion, and traffic [15]. This caching method causes a bottleneck in the origin server with an increase in the number of users. This method becomes less efficient because of the limited system resources of the cache server. Prefetching was proposed to solve this caching problem [35]. The prefetching method solves bottlenecks and traffic jams, and it allows the faster transmission of data. A proxy can effectively handle more user requests than caching, and it can help reduce the load on the origin server. However, one disadvantage of the prefetching method is that prefetched data may not be requested by the user. To solve this problem, a high-accuracy prediction model needs to be used [36]. Domènech et al. [35] conducted a study on indices-related prediction, resource usage, and latency evaluation related to prefetching.

3. Proposed Method

3.1. AR Prefetching

We proposed a prefetching method which partitions data. Excessive traffic occurs during the first connection if a user downloads all the AR data simultaneously. Excessive traffic increases network traffic, and latency increases owing to limited network bandwidth [37]. The latency of other users increases because of the increase in traffic. Figure 3 shows the differences between prefetching and conventional methods. Figure 3a shows the conventional method in which a load event occurs once, and the entire AR content is

prefetched simultaneously. Figure 3b shows the proposed method in which a load event occurs several times, and the predicted AR content is prefetched.

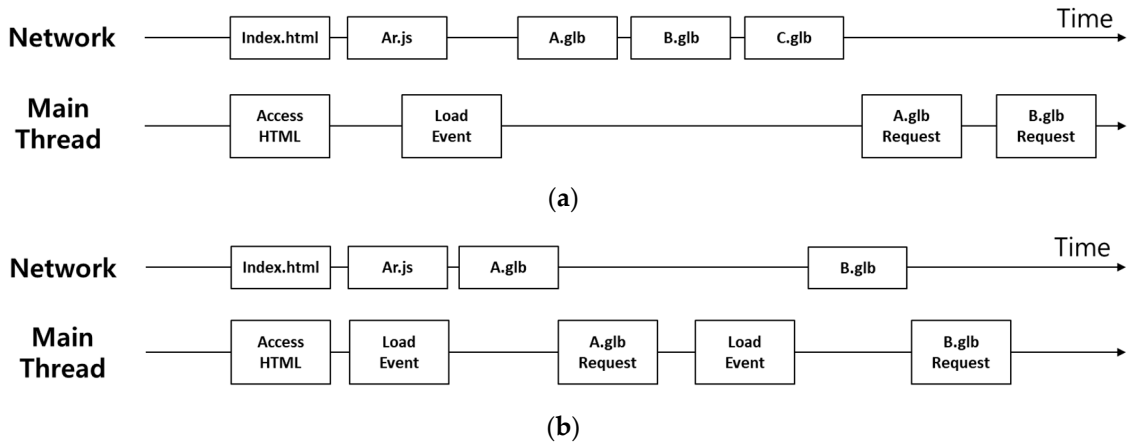


Figure 3. Resource loading sequence: (**a**) conventional method; (**b**) proposed method.

3.2. Client Server Architecture

Figure 4 shows the client and server architecture. This system consists of a client, web server, and database. The web server and database are located on edge servers.

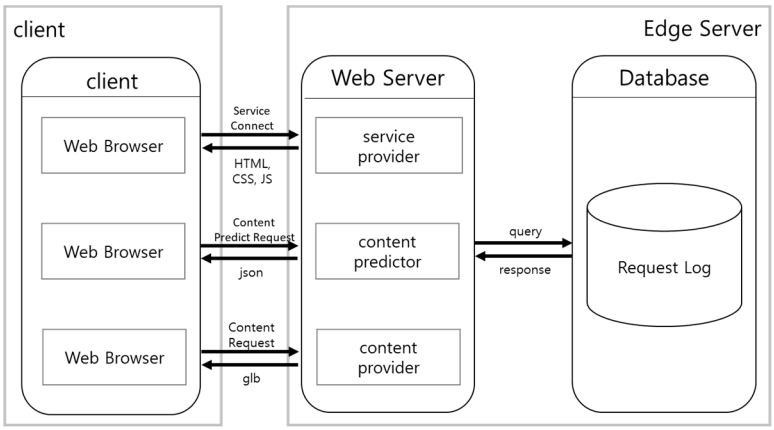


Figure 4. Client and server architecture.

3.2.1. Client

Figure 5 shows a flowchart of the client. The web browser sends a service access request to the web service provider and receives HTML, CSS, and JS to access AR services. When the AR is loaded, the client sends a request to the content predictor and initially receives the content information to be prefetched in JSON format. A list of content received from the content predictor is requested to the content provider and is prefetched. When the marker is recognized, a content output request is generated. If the content is prefetched, the content is provided immediately. However, if the content is not prefetched, the content is provided after prefetching the content. After providing the content, the list of content to be prefetched from the content predictor is received. If the list includes content that has not been prefetched, a request is sent to the content provider, and the content is prefetched.

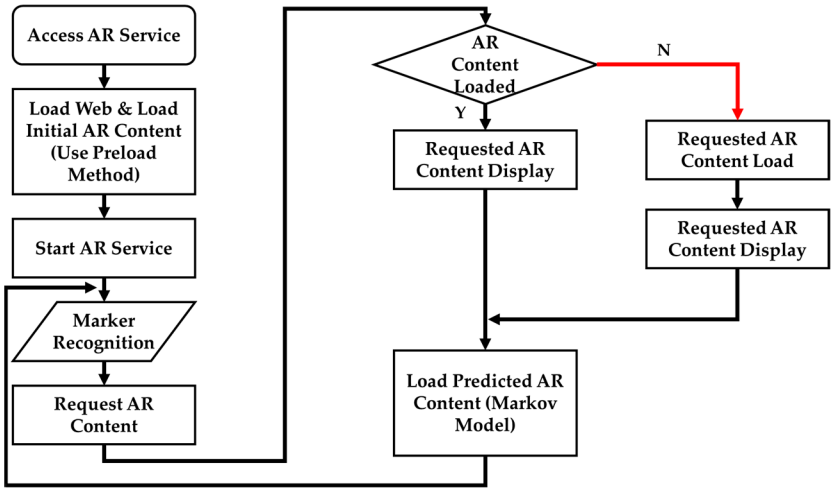


Figure 5. Client flowchart.

3.2.2. Edge Server

The edge server consists of a web server and database. The web server serves to transmit data to the client and consists of a service provider, content predictor, and content provider. The database stores request records.

The service provider provides a user interface to the client when the client accesses the AR service through a web browser. The information that the service provider provides to the client is in the form of HTML, CSS, and JS files, which compose the user interface.

The content predictor responds with a list of content to be prefetched in JSON format when a request for content prediction occurs in a web browser. Prefetching [34] based on the Markov model is used. The Markov model defines the transition probabilities between several states. The discrete-time Markov chain formula that predicts data is expressed as

$$\Pr(X_{n+1} = x | X_1, X2 = x_2, \ldots, X_n = x_n) = \Pr(X_{n+1} = x | X_n = x_n), \quad (1)$$

where n represents the order, and the state at n is defined as $x_1, x_2, \cdots, x_n$. Further, $X_1, X_2, \cdots, X_n$ represents the content. The formula for the calculation of statistical probability is

$$\check{P}(X(t+1) = x_c | X(n) = x_b) = \frac{n_{bc}}{\sum_{\alpha=a}^{0} n_{\alpha c}}, \quad (2)$$

where $X(t)$ denotes a random variable predicting AR content in time zone t; x_a, x_b, $\cdots$, x_o means $a \cdots o$ content; and $n_{\alpha c}$ represents the number of shifts from x_α to x_c content.

The content provider provides the corresponding 3D content as a glb file when a content prefetching request occurs in a web browser.

4. Experiments

4.1. Experimental Environment

For the experiment, an AR service was implemented in the "Nowon Career Center for Youth and Future" in Seoul, Republic of Korea, and the AR service data of users were collected. Figure 6 shows the mesh network access point (AP)/edge installation location and the AR image anchor location. In the experiment, 15 AR objects, each with a capacity of 10 MB to 10.5 MB, were used. The data were collected for three months from 1 June 2022 to 31 August 2022.

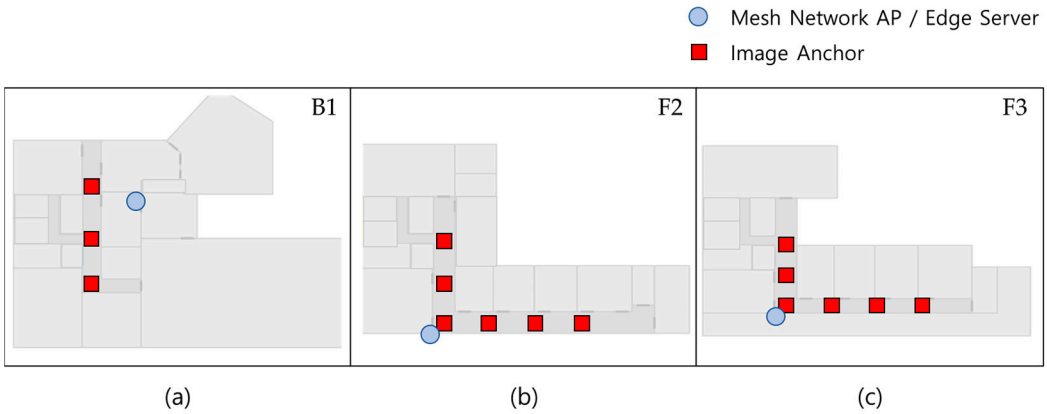


Figure 6. Test area drawings: (**a**) B1 drawing; (**b**) F2 drawing; (**c**) F3 drawing.

We placed mesh network AP, edge server, and anchor on building basement 1, floor 2, and floor 3. Figure 6 illustrates the test area drawings.

In the experiment, Samsung SM-T860 was used as the client device. The AP and edge server used VEEA's VHE10; it can communicate up to 300 Mbps using Wi-Fi-5-enabled devices. The AWS EC2 cloud server was used for an experimental comparison. AWS is the one of top cloud solution providers (CSPs), a pioneer, and the oldest cloud-service-providing company [38]. Table 1 lists the client information and Table 2 lists the server information used in this experiment.

Table 1. Client information used in the experiment.

Client (Samsung Tab S6 SM-T860)	
CPU	Qualcomm Snapdragon 855 SM8150 Platform
RAM	8 GB LPDDR4X SDRAM
OS	Android 11
Wi-Fi	Wi-Fi 1/2/3/4/5
Chrome	97.0.4692.98

Table 2. Server information used in the experiment.

	Edge Server (VEEA VHE10)	**Cloud Server (Amazon EC2)**
Type	-	t2.large
CPU	ARMv8 Quad Core processer, 1.5 GHz	2vCPU
RAM	8 GB DRAM	8 GB
Wi-Fi	Tri-band Wi-Fi5	-
NETWORK	1 Gbps	500 Mbps

In this experiment, we proposed a WMN edge server method for stably providing clients with low latency. The service area was expanded by installing an AP constituting each mesh with the IEEE 802.11 protocol. The WMN covered the entire AR service area using Wi-Fi. All APs were connected to each other in a mesh structure. The AP constituting the mesh amplified the WLAN signal and expanded the area; the edge server was configured in the same LAN layer as the client accessing the WMN. Further, it accessed the server only through communication with an internal network.

In the case of an edge server, network communication was performed using the internal network (LAN) without using the WAN and MAN; this reduced the effect of

reducing network-related variables. Figure 7 shows the network configuration of the WMN and edge server.

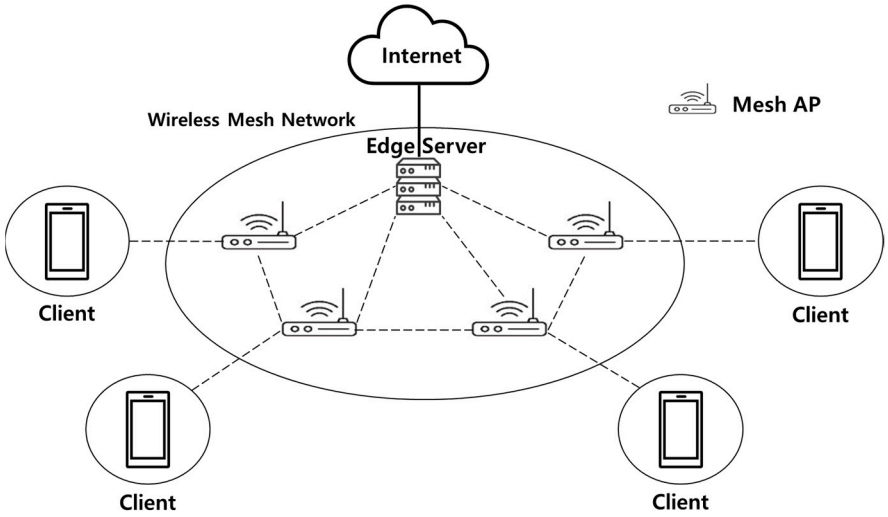


Figure 7. Wireless mesh network topology.

4.2. Evaluation Method

4.2.1. Hit Ratio

The hit ratio represents the ratio of the prefetch hits to the total number of objects requested by users. The formula for the *hit ratio* [39] of AR prefetching is expressed in Equation (3).

$$hit\ ratio\ (\%) = \left| \frac{\text{True}}{\text{True} + \text{False}} \right| \times 100 \tag{3}$$

where True represents the number of times a request occurs in the prefetching state, and False indicates the number of times the request has not been prefetched.

4.2.2. Waste Ratio

The waste ratio [40] refers to the ratio that has been prefetched; however, it is not used in the service. Equation (4) shows the *waste ratio* calculation formula.

$$waste\ ratio\ (\%) = \left| 1 - \frac{\text{Prefetch Hits}}{\text{Prefetchs}} \right| \times 100, \tag{4}$$

where prefetch represents the number of data points prefetched by the client. The prefetch hits represent the number of prefetches used.

4.2.3. Latency

Latency [39] is an index that measures the delay that occurs in a client. It is used to measure the initial latency when connecting and the latency of the display output when requesting specific content. Equation (5) shows the *latency* calculation formula.

$$latency(\text{ms}) = |T_1 - T_2|, \tag{5}$$

where T_1 represents completed times, and T_2 represents start times.

4.2.4. Moving Average

The moving average [41] is an indicator of the hit ratio, waste ratio, and latency trends. Equation (6) shows the moving average calculation formula.

$$\overline{x}_k = \frac{x_{k-n+1} + x_{k-n+2} + \cdots + x_k}{n} = \frac{1}{n}(x_{k-n+1} + x_{k-n+2} + \cdots + x_k), \quad (6)$$

where $x_1, x_{2,} x_3 \cdots x_k$ represents the $1st, 2nd, 3rd \cdots k$ data, and k represents the total amount of data. Further, n represents the average number of data points.

4.3. Experimental Results

In this study, the average of next content usage intermediate time was divided by the average prefetching time to obtain the number of prefetching AR objects. The average of next content usage intermediate time of the collected data was 1592.23 ms, and the average prefetching time was 385.91 ms.

4.3.1. Comparison between the Edge and Cloud Servers

The first experiment compared a cloud server and an edge server with a traditional AR service. The AWS cloud servers located in Singapore, Virginia, and Seoul were used in this experiment. The experiment measured the AR content loading times of the cloud and edge servers. The experiment was performed ten times, and the mean and standard deviation were calculated.

Figure 8 shows the load time of the AR service for each server. The number of attempts denotes the number times each experiment was performed. Table 3 lists the average latency and standard deviation of the data in Figure 8. The AR content load time was 4329.50 ms when the edge server was used; it exhibited faster load times than that of cloud servers. Further, the standard deviation was stable at 170.31 ms. The edge server reduced delays and was 14.49% faster than the cloud server. Further, stability increased by 4.54% when using the edge server, and this showed that latency and stability were excellent because of the reduction in nodes when the physical distance between the server and client was reduced.

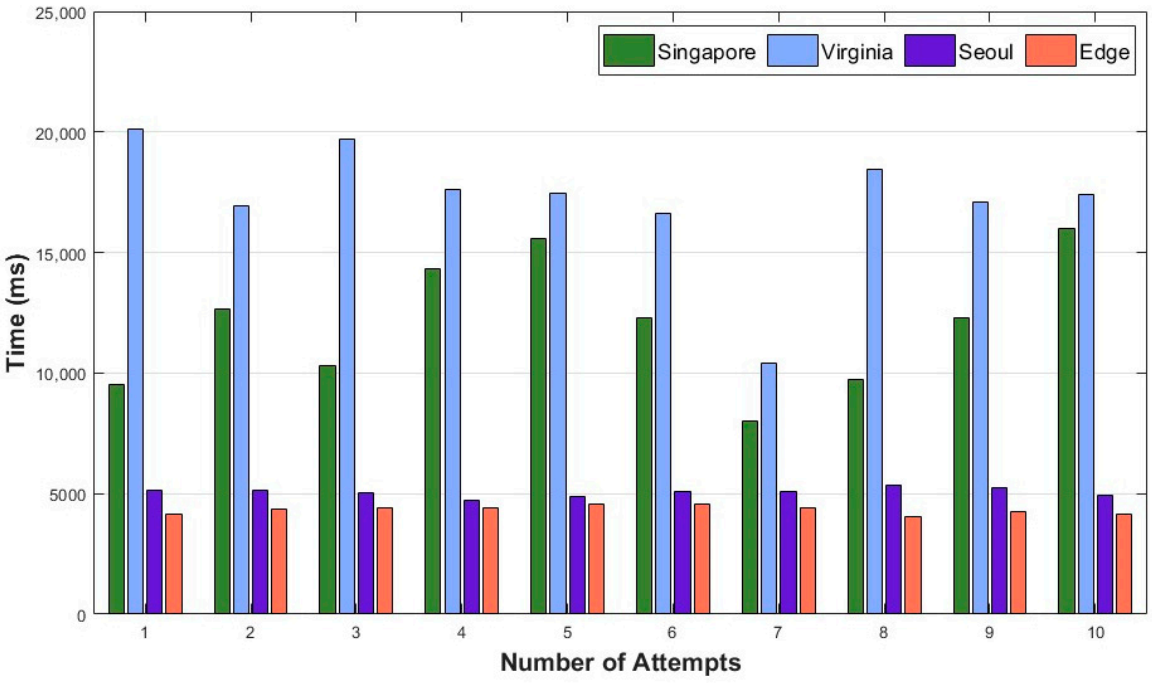


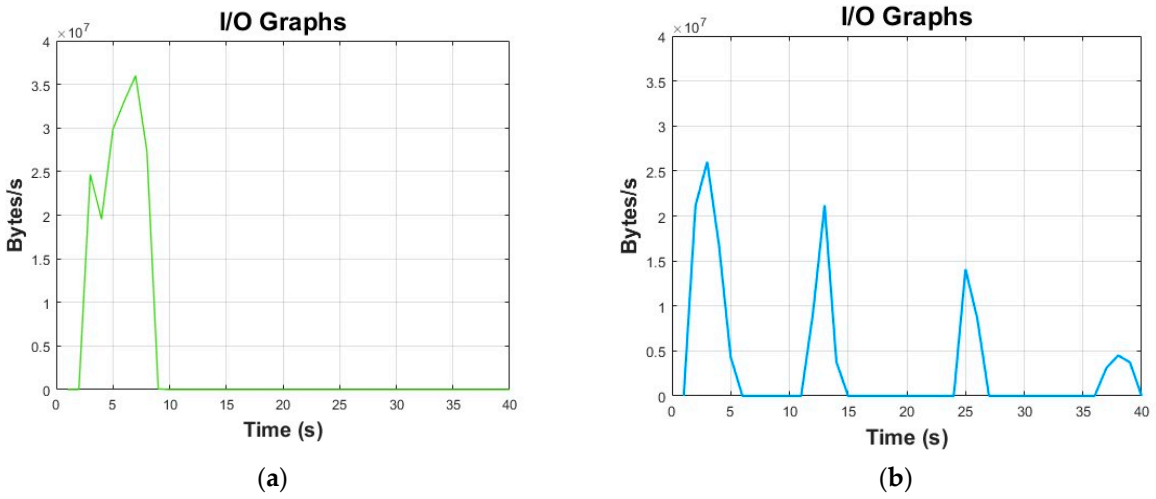
Figure 8. AR content load on the cloud server.

Table 3. Average and standard deviation of the AR content load on cloud servers.

	Average (ms)	Standard Deviation (ms)	Distance (km)
Singapore	12,070.50	2542.15	4688.87
Virginia	17,181.00	2506.36	11,137.60
Seoul	5062.90	178.41	11.34
Edge	4329.50	170.31	-

4.3.2. Comparative Analysis of the Network Traffic

Figure 9 shows the results of measuring traffic when users use AR content. Figures 9a and 8b show the traffic results of the conventional method and the traffic measurement results of the proposed method, respectively. In the conventional method, a relatively large amount of traffic was generated in the initial stage of accessing a web page. Compared with the conventional method, the proposed method was distributed and generated less initial traffic. In the proposed method, additional network traffic was generated by prefetching additional content during service use. The probability of content prefetching increased because of the increase in the number of requests. Therefore, additional traffic showed that the bytes per second decreased gradually with an increase in the request order.



(**a**) (**b**)

Figure 9. Network traffic I/O graph: (**a**) conventional method; (**b**) proposed method.

Table 4 shows Mean and standard deviation of the AR content loading of the proposed method. The proposed method showed superior results in terms of initial latency and total latency compared with the conventional method. False prefetching indicates that the corresponding content is not prefetched when a content request occurs. High latency occurs because the content is provided after fetching. True prefetching indicates that the corresponding content is prefetched when the content is requested. The prefetched content is provided immediately.

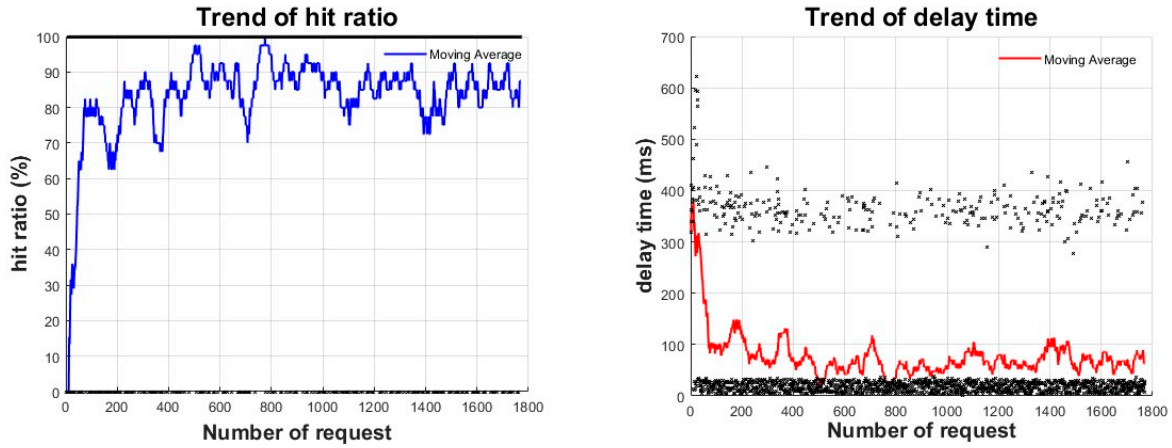
Table 4. Mean and standard deviation of the AR content loading of the proposed method.

	Average (ms)	Standard Deviation (ms)
Initial time	295.60	34.99
Prefetching false	385.91	60.08
Prefetching true	17.98	8.68

4.3.3. Results per Request of the Proposed Method

Figure 10 shows the results of content requests over time. Figure 10a shows the hit ratio; the points represent real data, and in the case of a linear graph, the trend of the

data is shown as a moving average, where the number of data points to be averaged is set to 30. The moving average of the hit ratio starts at 0% and increases as the user data collection progresses, which maintains it between 70% and 100%. Figure 10b shows the latency, which is inversely proportional to the hit ratio and shows a high delay time initially, which decreases to less than 100 ms.

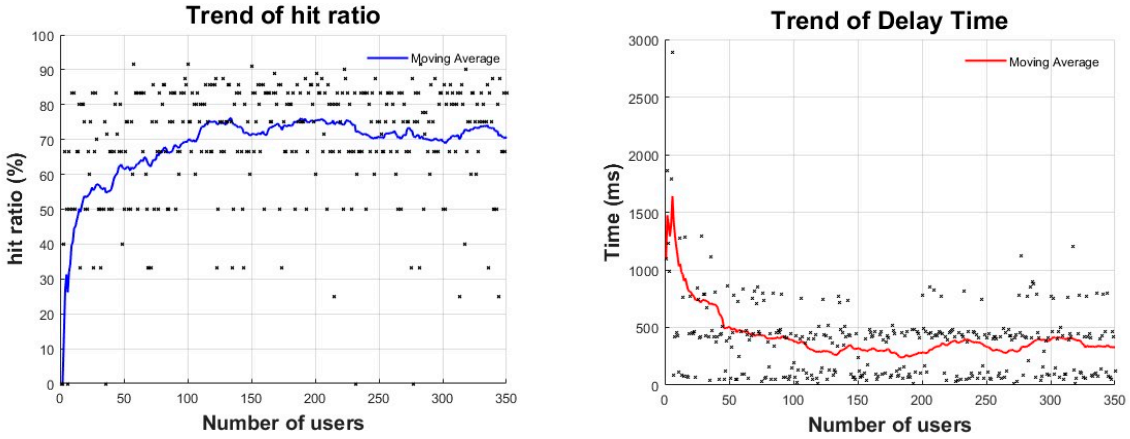


(**a**) (**b**)

Figure 10. Content request results in the chronological order of the proposed method: (**a**) hit ratio; (**b**) latency.

4.3.4. Average Result per User of the Proposed Method

Figure 11 shows the hit ratio and latency for each increase in user access. Figure 11a shows the average hit ratio for each user, and Figure 11b shows the total latency for each user. The total latency of a user was approximately 2000 ms. The total latency of a user also decreased to less than 400 ms with an increase in the hit ratio.



(**a**) (**b**)

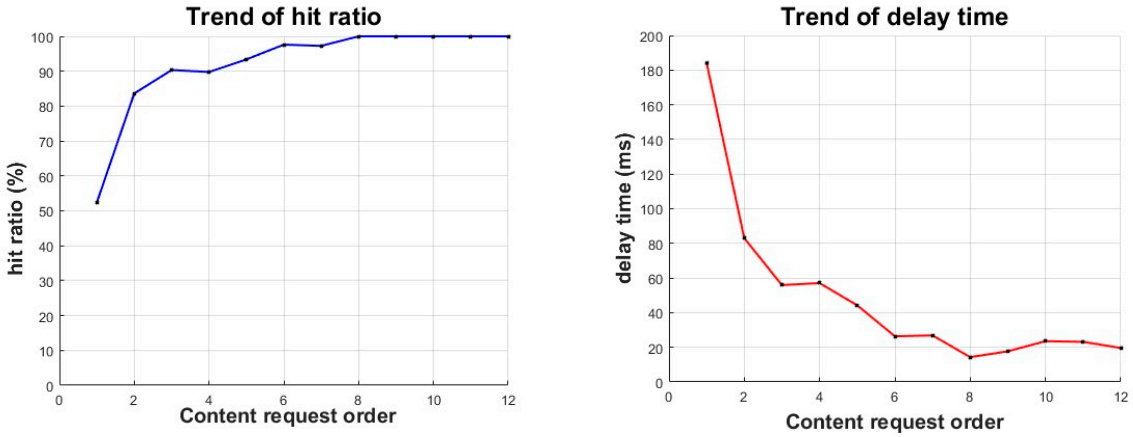
Figure 11. Average content request results of the connected users of the proposed method: (**a**) hit ratio; (**b**) total latency.

As user data accumulated, the hit ratio increased.

4.3.5. Results Based on Content Request Order of the Proposed Method

Figure 12 shows the hit ratio and latency for each content request order. Figure 12a shows the hit ratio. The first request per user had a low hit ratio of 50% because the experiment was conducted in an environment where the starting location was not determined.

From the second request onward, the hit ratio increased to more than 80%. Subsequently, it gradually increased and reached a hit ratio of 100% from the eighth request. Figure 12b shows the latency. In the case of content requested for the first time by each user, it took more than 180 ms, and this was relatively time-consuming. After the eigth request, when the hit ratio became 100%, the output time was 20 ms or less. The AR content for which prefetching was completed also increased because the number of requests per user increased. Even if the prediction failed, there was a high probability that the prefetching was completed. Therefore, the hit ratio converged to 100%.

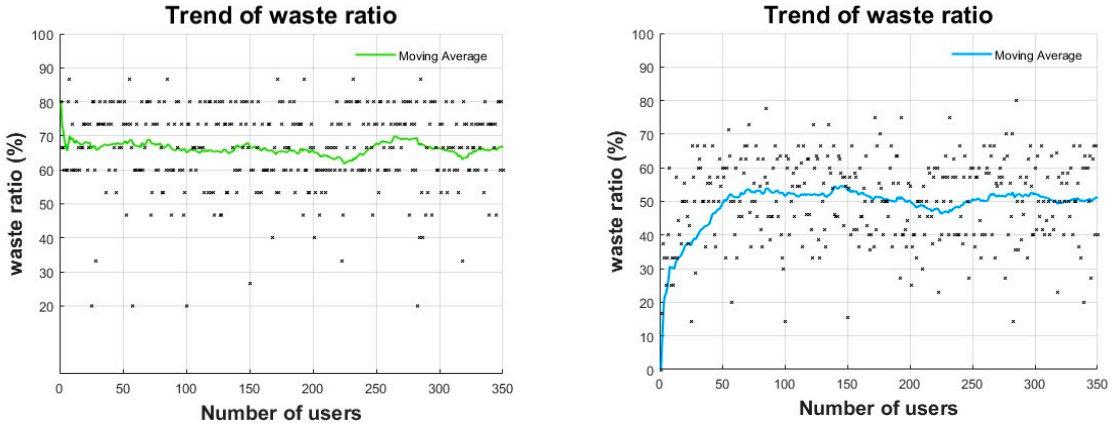


(**a**) (**b**)

Figure 12. Results of the content request order for the proposed method: (**a**) hit ratio; (**b**) latency.

4.3.6. Waste Ratio Results of the Proposed and Conventional Methods

Figure 13 shows the waste ratio that was prefetched but not used by the client. Figure 13a,b show the results of the conventional and proposed methods, respectively. The proposed method had an average waste ratio of 50.25%, and it was lower than that of the conventional method (66.28%). The proposed method initially exhibited a low waste ratio. Prefetching did not proceed because of the lack of user data and unpredictability. The amount of prefetching content increased as the user data accumulated, and this resulted in a waste ratio that increased to approximately 50% and was maintained.



(**a**) (**b**)

Figure 13. Result of waste ratio: (**a**) conventional methods; (**b**) proposed method.

5. Conclusions

This study provides a method for faster and more efficient WebAR services. Large amounts of 3D data require latency from the users. A method was proposed that reduces

latency in WebAR services by 81.5%. In the experiments, edge servers could physically reduce delays and were 14.49% faster than cloud servers. Further, the stability increased by 4.54%. The waste ratio related to unnecessary content prefetching was 16% lower than that of the conventional method. The Markov model was used to prefetch the predicted content rather than random prefetching to increase the hit ratio. The proposed method is more advantageous as it provides a large amount of AR content. Further, it is suitable for the realization of high-quality data in Web AR services. The results of this study have potential applications in areas such as AR games and AR docent services. A limitation of this study is that latency increased when prediction failed. Future research should increase the hit ratio by considering multiple parameters and advanced prediction algorithms. To this end, it is necessary to secure user data and apply machine learning.

Author Contributions: Conceptualization, S.C.; methodology, S.C.; software, S.C., S.H. and S.L.; investigation, S.H. and H.K.; writing—original draft preparation, S.C.; writing—review and editing, S.L. and S.K.; supervision, S.K.; project administration, S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (2020R1F1A1069079) and by Ministry of Culture, Sports and Tourism and Korea Creative Content Agency (Project Number: R2021040083).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author. The data are not publicly available because of privacy concerns.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Web XR Device API. Available online: https://www.w3.org/TR/webxr/ (accessed on 1 May 2022).
2. Three.js. Available online: https://threejs.org/ (accessed on 1 May 2022).
3. AR.js. Augmented Reality on the Web. Available online: https://ar-js-org.github.io/AR.js-Docs/ (accessed on 1 May 2022).
4. Mai, S.; Liu, Y. Implementation of Web AR applications with fog radio access networks based on openairinterface platform. In Proceedings of the 2019 5th International Conference on Control, Automation and Robotics (ICCAR), Beijing, China, 19–22 April 2019; IEEE Publications: Beijing, China, 2019; pp. 639–643. [CrossRef]
5. Han, D.-I.; Tom Dieck, M.C.; Jung, T. User experience model for augmented reality applications in urban heritage tourism. *J. Herit. Tour.* **2018**, *13*, 46–61. [CrossRef]
6. Naka, R.; Hagiwara, N.; Ohta, M. Accelerating data loading for photo-based augmented reality on web browser. In Proceedings of the 2018 IEEE 7th Global Conference Consumer Electronics (GCCE), Nara, Japan, 9–12 October 2018; IEEE Publications: Nara, Japan, 2018; pp. 698–699. [CrossRef]
7. Banerji, S.; Chowdhury, R.S. On IEEE 802.11: Wireless LAN technology. *IJMNCT* **2013**, *3*, 45–64. [CrossRef]
8. IEEE. 802.11TM Wireless Local Area Networks. Available online: https://www.ieee802.org/11 (accessed on 5 May 2022).
9. IEEE. P802.11 Wireless LANs Draft Terms and Definitions for 802.11s. Available online: https://mentor.ieee.org/802.11/dcn/04/11-04-0730-01-000s-draft-core-terms-and-definitions-802-11s.doc (accessed on 5 May 2022).
10. IEEE. 802.11-s Tutorial Overview of the Amendment for Wireless Local Area Mesh Networking. Available online: https://www.ieee802.org/802_tutorials/06-November/802.11s_Tutorial_r5.pdf (accessed on 5 May 2022).
11. Lin, Y.; Kemme, B.; Patino-Martinez, M.; Jimenez-Peris, R. Enhancing edge computing with database replication. In Proceedings of the 2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, China, 10–12 October 2007; IEEE Publications: Beijing, China, 2007; pp. 45–54. [CrossRef]
12. Seyun, C.; Woosung, S.; Sukjun, H.; Hoijun, K.; Seunghyun, L.; Soonchul, K. A Novle Method for Efficient Mobile AR Service in Edge Mesh Network. *Int. J. Internet Broadcast. Commun.* **2022**, *14*, 22–29.
13. Ren, J.; He, Y.; Huang, G.; Yu, G.; Cai, Y.; Zhang, Z. An edge-computing based architecture for mobile augmented reality. *IEEE Netw.* **2019**, *33*, 162–169. [CrossRef]
14. Xiuquan, Q.; Pei, R.; Schahram, D.; Ling, L.; Huadong, M.; Junliang, C.; Web, A.R. A promising future for mobile augmented reality—State of the art, challenges, and insights. *Proc. IEEE* **2019**, *107*, 651–666.
15. Ali, W.; Shamsuddin, S.M.; Ismail, A.S. A survey of web caching and prefetching. *Int. J. Adv. Soft Comput.* **2011**, *3*, 18–44.

16. Miyashita, T.; Meier, P.; Tachikawa, T.; Orlic, S.; Eble, T.; Scholz, V.; Gapel, A.; Gerl, O.; Arnaudov, S.; Lieberknecht, S. An augmented reality museum guide. In Proceedings of the 2008 7th IEEE/ACM International Symposium Mixed Augmented Reality, Cambridge, UK, 15–18 September 2008; IEEE Publications: Cambridge, UK, 2008; pp. 103–106. [CrossRef]
17. Lee, D.; Shim, W.; Lee, M.; Lee, S.; Jung, K.-D.; Kwon, S. Performance evaluation of ground AR anchor with WebXR device API. *Appl. Sci.* **2021**, *11*, 7877. [CrossRef]
18. Lee, G.A.; Dunser, A.; Kim, S.; Billinghurst, M. CityViewAR: A mobile outdoor AR application for city visualization. In Proceedings of the 2012 IEEE International Symposium Mixed Augmented Reality—Arts, Media, and Humanities (ISMAR-AMH), Atlanta, GA, USA, 5–8 November 2012; IEEE Publications: Atlanta, GA, USA, 2012; pp. 57–64.
19. Chung, N.; Han, H.; Joun, Y. Tourists' intention to visit a destination: The role of augmented reality (AR) application for a heritage site. *Comput. Hum. Behav.* **2015**, *50*, 588–599. [CrossRef]
20. He, J.; Ren, J.; Zhu, G.; Cai, S.; Chen, G. Mobile-based AR application helps to promote EFL children's vocabulary study. In Proceedings of the 2014 IEEE 14th International Conference on Advanced Learning Technologies, Athens, Greece, 7–10 July 2014; IEEE Publications: Athens, Greece, 2014; pp. 431–433. [CrossRef]
21. Nguyen, M.; Lai, M.P.; Le, H.; Yan, W.Q. A web-based augmented reality platform using pictorial QR code for educational purposes and beyond. In Proceedings of the 25th ACM Symposium on Virtual Reality Software Technology, Parramatta, Australia, 12–15 November 2019; ACM: Parramatta, NSW, Australia, 2019; pp. 1–2.
22. Barone Rodrigues, A.; Dias, D.R.C.; Martins, V.F.; Bressan, P.A.; de Paiva Guimarães, M. WebAR: A web-augmented reality-based authoring tool with experience API support for educational applications. In *International Conference on Universal Access in Human-Computer Interaction*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 118–128. [CrossRef]
23. Karthika, K.C. Wireless mesh network: A survey. In Proceedings of the 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 21–23 March 2016; IEEE Publications: Chennai, India, 2016; pp. 1966–1970. [CrossRef]
24. Passos, D.; Teixeira, D.V.; Muchaluat-Saade, D.C.; Magalhães, L.S.; Albuquerque, C. Mesh network performance measurements. In Proceedings of the International Information and Telecommunication Technologies Symposium (I2TS), Cuiabá, Brazil, 6–8 December 2006; pp. 48–55.
25. Navda, V.; Kashyap, A.; Das, S.R. Design and evaluation of IMesh: An infrastructure-mode wireless mesh network. In Proceedings of the Sixth IEEE International Symposium World of Wireless Mobile Multimedia Networks, Naxos, Italy, 13–16 June 2005; IEEE Publications: Taormina-Giardini Naxos, 2005; pp. 164–170. [CrossRef]
26. Paulon, J.V.M.; Olivieri de Souza, B.J.; Endler, M. Exploring data collection on Bluetooth mesh networks. *Ad Hoc Netw.* **2022**, *130*, 102809. [CrossRef]
27. Benyamina, D.; Hafid, A.; Gendreau, M. Wireless mesh networks design—A survey. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 299–310. [CrossRef]
28. Akyildiz, I.F.; Xudong, W. A survey on wireless mesh networks. *IEEE Commun. Mag. Inst. Electr. Electron. Eng.* **2005**, *43*, S23–S30. [CrossRef]
29. Loven, L.; Lahderanta, T.; Ruha, L.; Leppanen, T.; Peltonen, E.; Riekki, J.; Sillanpaa, M.J. Scaling up an edge server deployment. In Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Austin, TX, USA, 13–17 March 2020; IEEE Publications: Austin, TX, USA, 2020; pp. 1–7. [CrossRef]
30. Wang, S.; Zhao, Y.; Xu, J.; Yuan, J.; Hsu, C.-H. Edge server placement in mobile edge computing. *J. Parallel Distrib. Comput.* **2019**, *127*, 160–168. [CrossRef]
31. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
32. Premsankar, G.; Di Francesco, M.; Taleb, T. Edge computing for the internet of things: A case study. *IEEE Internet Things J.* **2018**, *5*, 1275–1284. [CrossRef]
33. Besson, E. Performance of TCP in a wide-area network: Influence of successive bottlenecks and exogenous traffic. In Proceedings of the Globecom'00—IEEE. Global Telecommunications Conference, Cat. No. 00CH37137, San Francisco, CA, USA, 27 November–1 December 2000; IEEE Publications: San Francisco, CA, USA, 2000; Volume 3, pp. 1798–1804.
34. Sukhmani, S.; Sadeghi, M.; Erol-Kantarci, M.; El Saddik, A. Edge caching and computing in 5G for mobile AR/VR and tactile internet. *IEEE Multimed.* **2019**, *26*, 21–30. [CrossRef]
35. Domènech, J.; Gil, J.A.; Sahuquillo, J.; Pont, A. Web prefetching performance metrics: A survey. *Perform. Eval.* **2006**, *63*, 988–1004. [CrossRef]
36. Pallis, G.; Vakali, A.; Pokorny, J. A clustering-based prefetching scheme on a Web cache environment. *Comput. Electr. Eng.* **2008**, *34*, 309–323. [CrossRef]
37. Robert, L.C.; Mark, E.C. Measuring bottleneck link speed in packet-switched networks. *Perform. Eval.* **1996**, *27–28*, 297–318.
38. Manish, S.; Tripathi, R.C. Cloud computing: Comparison and analysis of cloud service providers-AWs, Microsoft and Google. In Proceedings of the 9th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 4–5 December 2020; Volume 2020.
39. Christos, B.; Agisilaos, K.; Dionysios, K. Predictive Prefetching on the Web and Its Potential Impact in the Wide Area. *World Wide Web* **2004**, *7*, 143–179. [CrossRef]
40. Ibrahim, T.I.; Cheng-Zhong, X. Neural nets based predictive prefetching to tolerate WWW latency. In Proceedings of the 20th IEEE International Conference on Distributed Computing Systems, Taipei, Taiwan, 10–13 April 2000. [CrossRef]

41. Craig, A.E.; Simon, A.P. Is smarter better? A comparison of adaptive, and simple moving average trading strategies. *Res. Int. Bus. Financ.* **2005**, *19*, 399–411. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.