Hindawi
International Journal of Digital Multimedia Broadcasting
Volume 2022, Article ID 6214830, 11 pages
https://doi.org/10.1155/2022/6214830

Research Article

DASH Framework Using Machine Learning Techniques and Security Controls

Aref Shaheed and Haisam Al-radwan

Department of Telecommunication Engineering, Tishreen University, Latakia, Syria

Correspondence should be addressed to Aref Shaheed; aref_90718@svuonline.org

Received 22 March 2022; Revised 16 April 2022; Accepted 25 April 2022; Published 13 June 2022

Academic Editor: Alessandro Bruno

Interest in video streaming has increased recently, as it constitutes most of the traffic on the Internet and cellular networks. These networks use different video streaming technologies. One of the most famous technologies is DASH (which stands for Dynamic Adaptive Steaming using HTTP). DASH adapts streaming parameters according to network conditions and uses the HTTP protocol to communicate between the user and the server. DASH faces many challenges that may lead to video interruptions and poor quality of user experiences (QoE) such as bad network conditions and buffering level control. In addition to the lack of studies, we cover security issues for these types of services. In this paper, we proposed an integrated framework that consists of four components: quality prediction model, precache model, light web application firewall, and a monitoring system. These four components improve QoE and precache and increase the level of security. The results of the quality prediction model are used to predict the quality of the next segments depending on the user's network conditions and in the precache model to improve caching to reduce the load on the streaming system and rely more on cache servers. The proposed web application firewall is a light version used to defend against video streaming attacks and verify the existence of necessary HTTP headers. The quality predictor model with the generated dataset achieved 97% classification accuracy using DecisionTree, and this experiment proved the strong relationship between congestion periods and streaming quality, which is s the main key in QoE.

1. Introduction

1.1. Video Streaming Protocols. Video streaming technologies used four main protocols: HTTP 1.1, HTTP 2.0, RTMP, and RTSP [1].

HyperText Transfer Protocol (HTTP) is a protocol that works in the application layer, particularly in the webservers [2]. It depends on the TCP/IP protocol and is used to transfer data (HTML, files, and database query results). HTTP uses port 80 by default. System administrators can configure the webserver to work with other ports. HTTP is a standard for unifying the communication between clients and webservers; this protocol defines how to build a user request, how to send it to the server, and how the server responds to these requests. Three features make HTTP simple and effective: connectionless, media independent, and stateless.

The upgraded version of HTTP 1.0 and HTTP 1.1 is HTTP 2.0 [3]; this version has additional features compared to the old versions, as it is a binary protocol, multiplexed, and encrypted.

Real-Time Messaging Protocol (RTMP) is a protocol developed by Macromedia (now Adobe) and supported by Adobe Flash [4]. RTMP has various derivatives such as RTMPE (encrypted) [5], RTMPS (secure over SSL/TLS) [6], and RTMPT (encapsulated within HTTP requests) [7]. The packet loss in HTTP is slightly less than that in RTMP. It is preferable to use HTTP over RTMP in high fluctuation cases. Using RTMP is not recommended except in the case of small networks such as conferences and e-learning lectures.

Real-Time Streaming Protocol (RTSP) relies on the RTP protocol, the Real-time Transport Protocol, to transfer video segments to the clients [8]. A disadvantage of this protocol is the use of port 554; firewalls block this port by default.

1.2. DASH Technology. Adaptive Bitrate (ABR) streaming technology is designed to deliver and provide a consistent,

high-quality streaming experience to clients who use various types of devices in bad network condition situations (fluctuations) [9]. Segmented video streaming has become common for video content distribution over the Internet and mobile networks. DASH (stands for Dynamic Adaptive Steaming using HTTP) is a popular ABR algorithm [10].

DASH breaks the video file into segments and encodes these segments at different bitrates; typically, the length of every single segment is of 2 to 10 seconds. When the user requests a video file, one of the previously encoded files is sent depending on the current network conditions from the client side; a client requests the next segment by providing a file from the streaming server called "MPD," which stands for Media Presentation Description. This file contains media contents, available resolutions, and many useful details for DASH [11].

The goal of implementing DASH is to improve the QoE and to ensure that the video will keep on playing continuously adaptive to the network conditions to avoid any interruption if it occurred.

The bandwidth limitation, the delay between request and response, the rapid network condition fluctuation, and the client device limitations are the main key video streaming technology challenges [12]. These challenges make selecting the next appropriate resolution for the next segments and integrating with the client cache to avoid overflow and underflow not easy to achieve.

As the Internet and mobile networks are constantly expanding, expectations indicate that the traffic in mobile networks for video streaming applications will be about 80% of the total traffic in mobile networks at the beginning of 2020 and will increase annually [13].

Most recent works handled these key challenges by proposing a modified version of HTTP, streaming hardware architecture, or algorithms. In addition to the lack of studies that proposed a comprehensive framework that addressed all these issues mentioned in key challenges, in this paper, we proposed a secure and intelligent framework that uses machine learning techniques consisting of four components: quality prediction model, precache model, web application firewall, and monitoring system. These four components improve QoE and precache and increase the level of security.

The rest of this paper is organized as follows: Section 2 presents related works, Section 3 explains the proposed model, Section 4 discusses experiments, Section 5 contains detailed results and discussions (evaluation, comparison, and future works), and Section 6 includes the conclusion.

1.3. Machine Learning. Machine learning (ML) theory designs and analyzes algorithms that allow computers to "learn" automatically and conduct human tasks rather than coding logic in traditional programming languages [14]. Machine learning algorithms can analyze and automatically obtain rules from data and use these rules to make decisions for new cases. Machine learning tasks are divided into three sections: classification, association, and clustering [15].

1.4. Web Application Firewalls. OWASP is a popular foundation that works to improve the security of software through various projects and guides. [16] defined web application firewall (WAF) as follows: "web application firewall (WAF)" is an application firewall for HTTP applications. It applies a set of rules to an HTTP conversation. Generally, these rules cover common attacks such as Cross-site Scripting (XSS) [17] and SQL Injection [18]. WAF exists between the client and the server, it verifies received a request, and depending on the classification results, WAF either forwards the request to the webserver or drops it [19].

2. Related Works

We can categorize related works into three categories:

2.1. Modify HTTP Structure. These studies proposed a modified version of HTTP. HTTP 1.1 is used as the default version in DASH. Alhamad and Kazi [20] proposed a light HTTP version to produce smaller requests and responses. This proposed version enhances QoE significantly and decreases the delay between requests and responses. Modifying the widely used protocol structure, HTTP requires modifying the current software and devices. It is an expensive and impractical solution.

Wei et al. [21] proposed using HTTP 2.0 instead of HTTP 1.1 to take advantage of HTTP 2.0 features, which enhance QoE. This version of the protocol is unpopular as most software and devices use HTTP 1.1.

2.2. Extend or Modification Architecture. These studies proposed hardware extensions in the streaming architecture. Qiao and He [22] proposed extending the streaming architecture with proactive precaching servers in base stations to enhance QoE as the client can request cached segments from these servers with the client handover from the base station to another base station. On the other side, this approach is costly as it requires the setup of a proactive cache server at every single base station and the precaching process needs effective management to avoid consuming power and storage.

Bruneau-Queyreix et al. [23] proposed to assist in responding to requested cached segments by providing adjacent base stations. This study inexpensively enhances precaching as it does not require setting up any additional server. However, it should face the challenge of consuming resources in adjacent base stations.

Al-Habashna et al. [24] proposed to assist precaching in the base station with the clients that centered the cluster called storage members (SM). This study is more effective in the cost term and decreased delay between requests and responses but at the expense of storage members QoE, who are also clients.

Qiao et al. [25] proposed a double-buffer system that guarantees better QoE for clients on highway, a buffer system applied at every base station and every vehicle (client on the highway). This study suggests storing the content of requested video segments in all base stations. Precaching in this proposed model enhances QoE at the expense of resources.

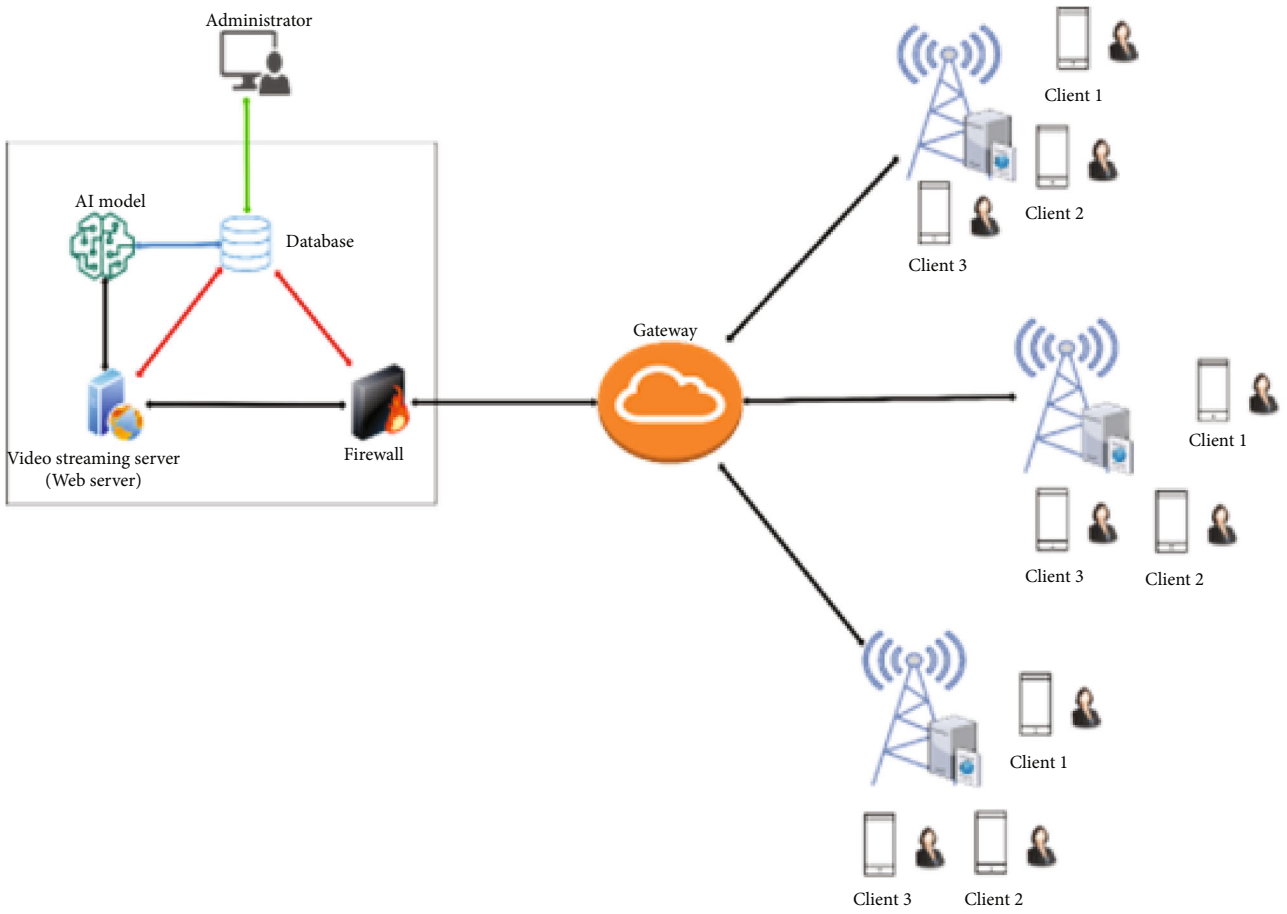

Figure 1: Proposed framework architecture for mobile networks.

Input: web logs
Output: The quality predictor model
1 **Start**
2 Configure the web service to satisfy framework requirements
2–1 Configuration of storing additional details while logging requests
2–2 Configuration of removing redundant HTTP headers
2–3 Configuration of forcing add and checking security HTTP headers
3 Collecting web logs
4 Process the web logs to generate the dataset
5 Select the appropriate classifier for the Quality predictor model
6 Training generated dataset using the selected classifier
7 **End**

Algorithm 1: Quality predictor model implementation.

Host - - [Time] "HTTP_method domain/resource HTTP_version" Status_code
192.168.10.250 - - [07/Jul/2021:22:17:39 +0300] "GET /bbb_30fps_320x180_200k_0.m4v HTTP/1.1" 200

Host - - [Time] "HTTP_method domain/resource?buffer_level=x HTTP_version" Status_code Response_size
192.168.10.250 - - [07/Jul/2021:22:17:39 +0300] "GET /bbb_30fps_320x180_200k_0.m4v?bufferLevel=59.32 HTTP/1.1" 200 705

Figure 2: Default web service log and new log after configuring the web service with examples.

Figure 3: Request and response headers after requesting a segment and receiving it.

Table 1: Features extracted from the log requests of the web service.

Feature	Type	Description
Time	Input	The current time (hour only, value range [0, 23]
Type	Type	Type
Size	Input	Response size
Buffer	Input	The current buffer level of the client
Bitrate	Output	The current bitrate of the client (value range [1, 5]

2.3. Enhance Algorithms and Use Machine Learning. These studies proposed dedicated algorithms to enhance QoE, reduce delay, and decrease segment size in different manners. Some of these studies use artificial intelligence techniques, and others focus on enhancing the performance of a dedicated component in the streaming architecture. These studies contributed significantly to enhancing video streaming services, but they need to be deployed in an integrated framework to strengthen their contributions.

Claeys et al. [26] proposed using Q-Learning on the client-side to enable clients to learn with tunable reward; this approach will adapt streaming behavior dynamically depending on network conditions to maximize QoE.

Table 2: Categorized bitrate levels.

Bitrate	Level
25	1
50	
76	2
101	
125	3
188	
313	4
495	
991	5
1493	

Hu and Cao [27] proposed an Energy-aware CPU Frequency Scaling (EFS) algorithm that reduces the total power consumed by the CPU. The low CPU frequency reduces the CPU energy but increases the data transmission time and then increases the energy consumption and vice versa. This algorithm achieved a balance between CPU frequency and transmission time, but it did not consider the performance in cases of high fluctuations in network conditions.

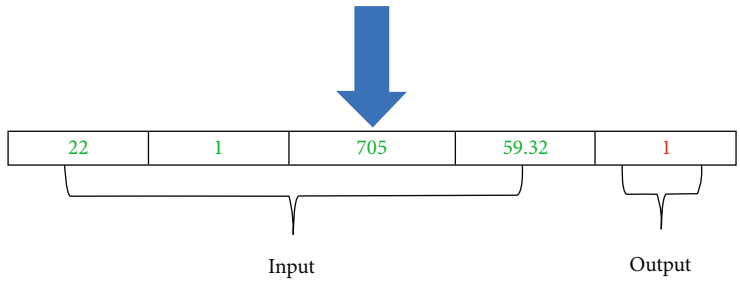

Figure 4: Dataset row after parsing the raw web service log request.

Polakovič et al. [28] proposed a saliency-based extension of the DASH video delivery system, and it is useful for enhancing QoE in bad network conditions by preserving more bits to the important part of the frame rather than allocating bits to the whole frame.

Yousef et al. [29] used machine learning techniques with streaming information (buffer level, bandwidth, previous bandwidth, download time, and previous bitrate) to predict the behavior of ABR algorithms. They conducted experiments on six different ABR algorithms BBA, BOLA, CONVENTIONAL, PANDA, FESTIVE, and Robust MPC using six different machine learning classifiers: Logistic Regression (LGS), Support Vector Machine (SVM), Random Forest (RF), DecisionTree (DT), Ada Boost (AdBst), Gradient Boost (GrdBst), Naive Bayes (NB), and K-Nearest Neighbours (KNN). Experiments show that both Random Forest and Gradient Boosting achieved very high prediction accuracy among other used ML classifiers.

3. Proposed Framework

Our proposed framework consists of four major components: quality prediction model, precache model, light web application firewall, and a monitoring system. These components work comprehensively to mitigate the impact of the key challenges of DASH. We maintained the traditional video streaming architecture without adding any costly resources or modifying the process. Instead, only one server was used for proactive prediction using machine learning techniques in the streaming service data center. A quality predictor model was deployed on this server to avoid consuming streaming service resources (see Figure 1).

3.1. Quality Prediction Model. This component achieves two goals and predicts the next appropriate segment while the client plays a video and assists the precache model to manage the precaching process more effectively (see Algorithm 1). In this section, we will discuss "configuring web service for satisfying framework requirements" and processing web service logs to generate the dataset. The training model will be discussed in Section 4. Configuring the web service includes three basic steps to configure a video streaming server to work with the proposed framework. Streaming service implemented through web service (e.g., Apache or Nginx), web service logs every single request arrived, and these logs are used to generate a dataset that is used in the training quality predictor model. Usually, the log requests of the web service have a default format. The system administrator must customize this format to fit the proposed solution. The default format used in logging requests needs customization to have additional features that will be included in the dataset (step 2–1 in Algorithm 1). Also, an additional configuration is needed to remove redundant headers sent from the web server to the client; this would help a lot in reducing responses sent to the client. For protection, also web service needs more configuration to force adding security HTTP headers.

The default log record contains the following information: host, time, HTTP method, requested resource (in video streaming case, it is the requested segment), HTTP version, and HTTP status code.

We customized the client player to send "buffer_level" in every single request, which describes the current buffer level in the client player, and the web service to store the response size in logs in addition to the default information (see Figure 2).

```
   Input: HTTP request "r"
   Output: HTTP response
1     Start
2     while WAF is "ON"
3       HTTP request "r" received
4       If "buffer_level" exists in the payload.
5         if "buffer_level" parameter value is an integer
6           if security headers exist
7             if IP is not on the blacklist
8               Pass request to streaming service
9             else
10              drop request and block the IP
11            endif
12          else
13            drop request and block the IP
14          endif
15        else
16          drop request and block the IP
17        endif
18      else
19        drop request and block the IP
20      endif
21    End
```

Algorithm 2: Light WAF methodology.

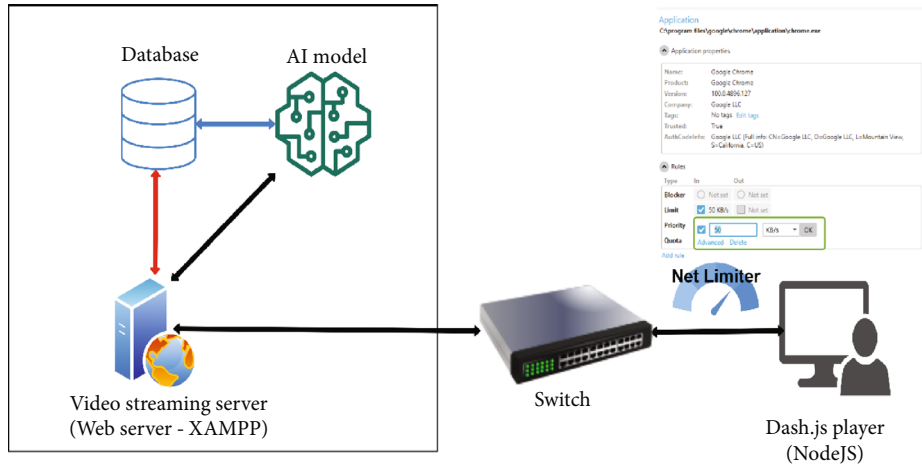

Figure 5: Experiment environment architecture.

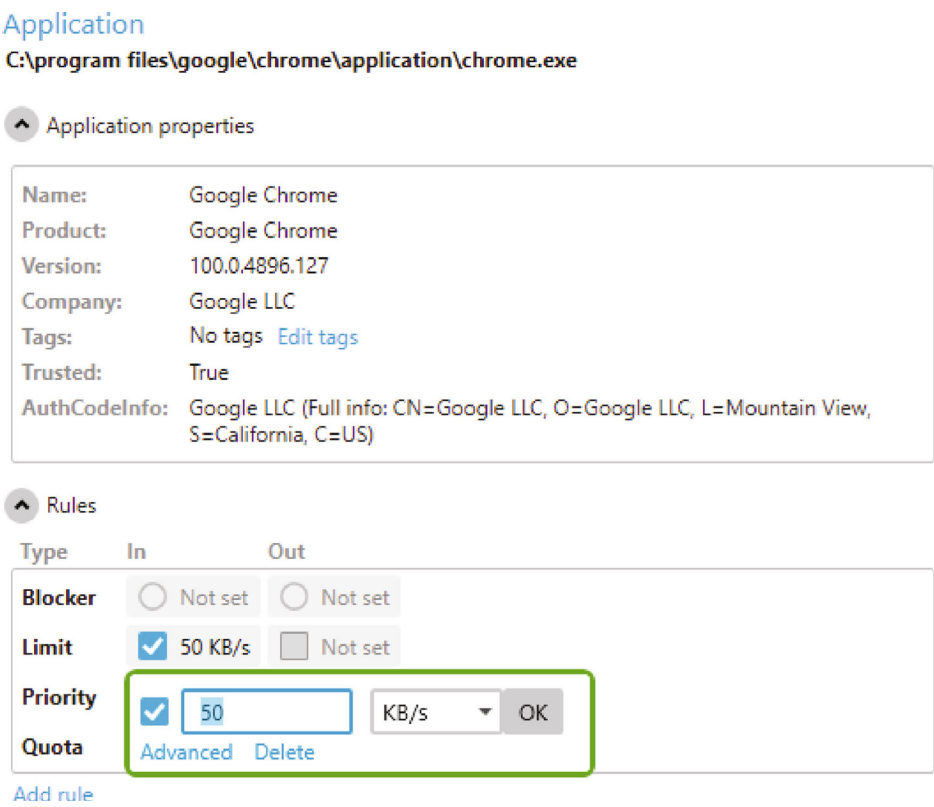

Figure 6: Net Limiter panel to control browser bandwidth.

Instead of making the client send the current bitrate, we can extract it from the segment name requested by the client. Finally, after the web service is configured, logs contain the following information: host, time, HTTP method, requested resource (in video streaming case it is the requested segment), current client buffer level, current client bitrate, response size, HTTP version, and HTTP status code. This information will be used to generate the dataset.

The next step is to configure the web service to remove redundant headers that will not affect the streaming service and increase response size (step 2–2 in Algorithm 1). Additionally, the client player should reduce the request size. The main goal of this step is to reduce request and response sizes to contain only useful and needed information in streaming services. Informational headers will not be sent to the client as they consume the total size. The attackers take advantage of these headers in the information-gathering phase [30], e.g., the "Server" header and "Access-control-Allow-Methods" header (see Figure 3; these headers are highlighted in black border).

Finally, the web service and client player need to be configured to force using security headers needed to increase the

```
    Input: HTTP requests by playing video many times
    Output: Apache logs
1       Start
2       Initiate test_hours = [2, 6, 10, 14, 20, 23]
3       Initiate test_bandwidths = [10mbps, 2mbps, 1mbps, 800kbps, 600kbps, 200kbps, 100kbps]
4       Initiate i =0
5         While loop_counter < length(test_hours)
6           set web server clock to test_hours[i]
7           set browser bandwidth to test_bandwidths[i]
8           Play video with dash.js
9           i = i +1
10        Endwhile
11      End
```

Algorithm 3: Generating dataset.

security level of the streaming service process and prevent common attacks [31] (step 2–3 in Algorithm 1), e.g. "Content-Security-Policy," "Strict-Transport-Security," and "Content-Security-Policy" headers [32].

After configuring the web service, the dataset can be generated using web service logs. The dataset contains the following information (see Table 1): buffer level, time (extracted hour only from DateTime string), response size, type of segment (video or audio), and bitrate (bitrate categorized to simplify the problem) (see Table 2).

Every single raw request in a row will be converted to a table row that contains five fields (4 input features and one output class). The quality predictor model will get parsed requests, extract features, and predict the appropriate bitrate depending on inputs (see Figure 4).

We conducted the experiments on the generated dataset with four fast and popular classifiers in real-time applications (Linear SVC, KNeighbours, Naïve Bayes, and DecisionTree) [33], and we selected DecisionTree. This classifier achieved the highest accuracy and has the best big O time complexity (it will be discussed in detail in Section 5).

3.2. Precache Model. Precaching works before congestion periods and depends on the results of the quality prediction model. We proposed a mathematical coefficient to evaluate the effectiveness of precaching based on the number of requests received by the webserver and the number of requests received by the cache server (see Equation (1)).

A new dataset of video streaming statistics can be used with the quality predictor model and clustering algorithms to precache effectively. The proposed mathematical coefficient called the "precaching effectiveness coefficient" will evaluate the model and help to enhance it.

$$E = \frac{\text{sr}}{\text{tr}} \times 100, \tag{1}$$

E is the precaching effectiveness coefficient, sr are requests received by web service, cr are requests arrived at cache server, and tr are total requests received by video streaming service.

Table 3: Big O time complexity of classifiers.

Classifier	Complexity
DecisionTree	$O(p)$ p: the number of features
SVM (kernel)	$O(nsv \times p)$ p: the number of features nsv: the number of support vectors
K-Nearest neighbours	$O(n \times p)$ n: the number of training samples p: the number of features
Naive Bayes	$O(p)$ p: the number of features

Total requests received by the video streaming service are calculated as follows:

$$\text{tr} = \text{sr} + \text{cr}. \tag{2}$$

For example, if the total requests received by the system are 1000 requests, the cache server responds to 430 requests and forwards the rest to the streaming server. The precaching effectiveness coefficient was calculated as follows:

$$\text{sr} = 1000 - 430 = 570, \tag{3}$$

$$E = \frac{570}{1000} \times 100 = 57\%. \tag{4}$$

3.3. The Light Web Application Firewall. It is a customized web application firewall that configures its rules and behavior to detect video streaming attacks by checking only the necessary HTTP headers, instead of deploying a full web application firewall. The main goal is to enhance performance without affecting the security level.

As video streaming technologies use HTTP, they are subject to web attacks, and it requires web security controls to increase the security level and mitigate these attacks.

OWASP listed the top 10 attacks [34] for web applications and services in a project called "OWASP Top 10." The last release of this project was published in September 2021.

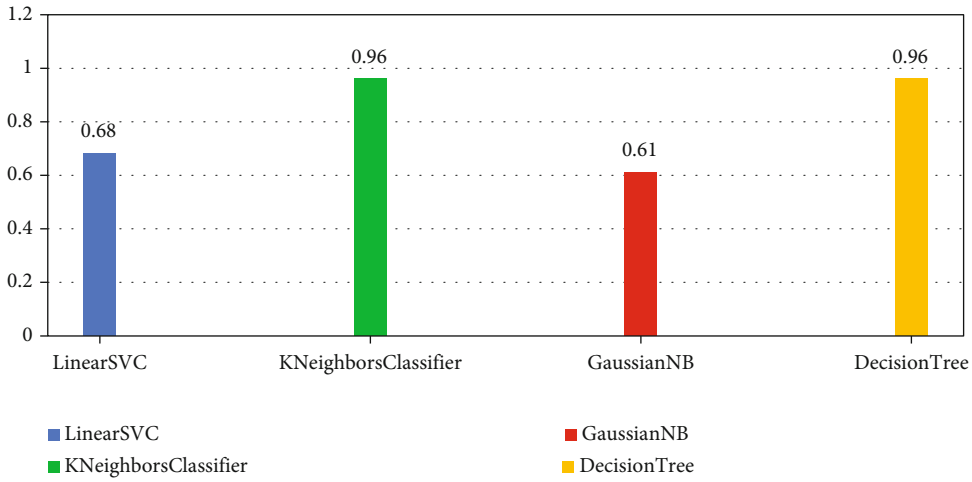

Figure 7: Results of training of four classifiers with the generated dataset.

We prefer to deploy light WAF instances in cache servers, as it is the first to face potential attackers. It receives requests and either responds to the customer or forwards it to the streaming server.

The proposed light WAF checks the existence of security headers and check payloads; it must contain only the "buffer_level" parameter as an integer value. It will drop any request unformed in this pattern and block the user immediately (see Algorithm 2).

3.4. Monitoring System. The monitoring system is a web application that helps technical administrators monitor the entire streaming process. This web application is connected to a database to retrieve all needed information.

4. Experiments

This section discusses the technical environment of the streaming server and the client player to generate a dataset by playing a video: parsing the web logs to generate the dataset, select a classifier, and train.

4.1. Technical Environment. We built a local web environment to generate the web logs by playing video many times at different hours of the day to fill webserver logs and generate enough rows to get a dataset.

The technical environment consists of the following components: a webserver (Apache 2.4), web client (Dash.js), Google Chrome browser, Net Limiter application to control browser bandwidth, and Python to write necessary scripts (see Figure 5).

The webserver configured to store requests in logs; as mentioned in Section 2.1, we modified the dash.js to send "buffer_level." We deployed a segmented "Big Buck Bunny" [35] video on the webserver with different bitrates.

4.2. Generating the Dataset. The client (dash.js) should request video many times in different hours of the day (by modifying webserver hour) with different network conditions (network conditions simulated by set bandwidth Net Limiter; see Figure 6) to fill the webserver logs (see Algorithm 3). Then, a python script will parse these logs and convert them to a CSV file that contains five columns (four input features and one output class).

Table 4: Classification precision of DecisionTree for bitrate classes.

Precision	Bitrate class
100%	1
99%	2
92%	3
76%	4
93%	5

This algorithm generates about 2000 HTTP requests in the webserver logs, hours, and bandwidth which are set depending on a private ISP database that holds detailed sessions for 10,000 clients between 2015 and 2018.

4.3. Selecting Classifier and Training. We trained the generated dataset with four classifiers (Linear SVC, KNeighbours, Naïve Bayes, and DecisionTree). Previous classifiers are popular in real-time applications. Depending on the training results, we selected DecisionTree to be used in deployed quality predict model (it will be discussed in detail in Section 5).

5. Results and Discussion

This section discusses the results of the experiments, the limitations of related works and how the proposed framework addresses these limitations, and the future works.

5.1. Experiment Evaluation. The generated dataset was tested with four classifiers: Naïve Bayes, KNeighbours, Linear SVC, and DecisionTree.

Evaluation keys for selecting one classifier from these four classifiers are classification accuracy, big O time complexity in training, testing, and execution (when the classifier is deployed to work in a real environment), and efficiency in real-time applications.

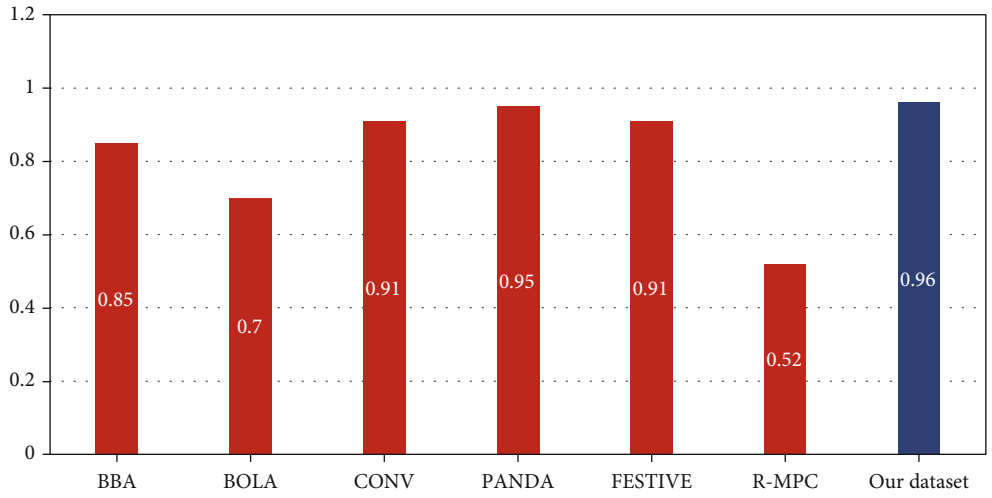

Figure 8: Comparison of the six datasets training results using KNN in the study [29] and our generated dataset training results using KNN.

Table 5: Limitations of related work approaches and how the proposed framework addressed it.

Related works approach	Contribution and limitation	Proposed framework approach
Modifying HTTP version	+ Reduce requests and response size - Modifying HTTP requires modifying all applications and devices, and it is not a practical solution as HTTP is widely spread	+ Removing redundant headers
Extending or modifying the architecture	+ Enhance QoE - Consuming power and costly	+ Precache model
Enhance algorithms and use machine learning	+ Enhance QoE + Preserve resources - Used to predict segments from the client side - Not deployed in an integrated framework	+ Used in prediction and precaching models

All previous classifiers are popular in real-time applications; the differences exist in big O time complexity [36] (see Table 3) and classification accuracy (see Figure 7).

Depending on the results, we choose DecisionTree as it achieved the highest classification accuracy rate (see Table 4) and the lowest Big O time complexity and is efficient with real-time applications.

5.2. Benchmarking with the Related Work. Most of the discussed related work in Section 2 proposed a modification to the structure itself. We can compare the prediction model with the proposed approach by Yousef et al. [29], who trained six datasets using different classifiers. Figure 8 describes a comparison of the six datasets training results using KNN in the study [29] and our generated dataset training results using KNN. We chose KNN in this comparison depending on the classifier selection results of Section 5.1.

5.3. Limitation of Related Work and Contributions of the Proposed Framework. Related studies focused on partially enhancing instead of proposing a comprehensive framework to address all key challenges of video streaming technologies. This section briefly discusses the main approaches that researchers have used to improve video streaming techniques, their effects, and how we have overcome them in the proposed framework (see Table 5).

5.4. Future Work

(i) Propose additional features for the quality predictor model

(ii) Use a new dataset (generated or from ISPs) for the quality predictor model

(iii) Build a clustering model for the precaching model

(iv) Study video streaming attacks to extend and enhance the proposed light WAF

(v) Use saliency detection in a saliency-based extension. We recommend the deep learning approach proposed by Wang et al. [37].

(vi) Extend the framework with new components

6. Conclusion

In this paper, we proposed an integrated framework for video streaming over HTTP; this framework handles major issues in this technology (security, caching, and quality of

service). We described the entire framework, focused on detailing the first component, and proposed bold lines for other components. This framework consists of four components: quality prediction model, precache model, light web application firewall, and a monitoring system. These four components improve QoE and precache and increase the level of security.

The streaming service architecture is not costly—only one server with a streaming server—and the configuration is simple enough for client player and server side. The quality predictor model extracts features from the web service log to predict appropriate streaming quality depending on client network conditions; these features are buffer level, time, response size, type of segment, and bitrate. This model achieved 97% classification accuracy using DecisionTree with the generated dataset. The results of this component were used in the precache model to assist precaching, and we proposed a mathematical coefficient to evaluate the precache model. A light WAF is proposed to protect streaming services with custom and effective rules—checking the existence of certain headers and payloads—instead of implementing a full and heavy WAF. Additionally, a monitoring component contains a web application to monitor the entire process.

Data Availability

The generated dataset with Python scripts will be available in the following repository: https://github.com/aref2008/dash_framework.

Conflicts of Interest

There are no conflicts of interest to declare.

Acknowledgments

We would like to thank everyone who provided any support to complete this work.

References

[1] A. Aloman, A. I. Ispas, P. Ciotirnae, R. Sanchez-Iborra, and M.-D. Cano, "Performance evaluation of video streaming using MPEG DASH, RTSP, and RTMP in mobile networks," in *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 144–151, Munich, Germany, 2015.

[2] R. Fielding, J. Gettys, J. Mogul et al., *Hypertext transfer protocol–HTTP/1.1*, 1999.

[3] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F. De Turck, "An HTTP/2 push-based approach for low-latency live streaming with super-short segments," *Journal of Network and Systems Management*, vol. 26, no. 1, pp. 51–78, 2018.

[4] X. Lei, X. Jiang, and C. Wang, "Design and implementation of streaming media processing software based on RTMP," in *2012 5th International Congress on Image and Signal Processing*, pp. 192–196, Chongqing, China, 2012.

[5] M. Kim and B. An, "Video contents security streaming," *The Journal of The Institute of Internet, Broadcasting and Communication*, vol. 12, no. 5, pp. 67–74, 2012.

[6] O. E. Osuagwu, C. Chinwe Ndigwe, U. S. Ihedigbo, and O. Babatunde, "Video conferencing: most effective technology to run assemblies and meetings for large audience dispersed in distant locations: is it feasible to deploy in Nigeria?," *West African Journal of Industrial and Academic Research*, vol. 17, no. 1, pp. 73–81, 2017.

[7] S. Laine and I. Hakala, "H. 264 QoS and application performance with different streaming protocols," *EAI Endorsed Transactions on Future Intelligent Educational Environments 2015*, 2015.

[8] H. Schulzrinne, A. Rao, and R. Lanphier, *Real Time Streaming Protocol (RTSP)*, 1998.

[9] Y. Sani, A. Mauthe, and C. Edwards, "Adaptive bitrate selection: a survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2985–3014, 2017.

[10] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[11] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 78–85, 2012.

[12] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid, "Quality of experience for streaming services: measurements, challenges and insights," *IEEE Access*, vol. 8, pp. 13341–13361, 2020.

[13] A. Hodroj, M. Ibrahim, and Y. Hadjadj-Aoul, "A survey on video streaming in multipath and multihomed overlay networks," *IEEE Access*, vol. 9, pp. 66816–66828, 2021.

[14] M. I. Jordan and T. M. Mitchell, "Machine learning: trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[15] D. Sharma and N. Kumar, "A review on machine learning algorithms, tasks and applications," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 6, no. 10, pp. 1548–1552, 2017.

[16] Owasp, *About Us|The OWASP Foundation*, https://owasp.org/about/.

[17] G. Wassermann and S. Zhendong, "Static detection of cross-site scripting vulnerabilities," in *2008 ACM/IEEE 30th International Conference on Software Engineering*, pp. 171–180, Leipzig, Germany, 2008.

[18] Z. S. Alwan and M. F. Younis, "Detection and prevention of SQL injection attack: a survey," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 8, pp. 5–17, 2017.

[19] H.-Y. Lee and H.-S. Yang, "Construction of security evaluation criteria for web application firewall," *Journal of digital Convergence*, vol. 15, no. 5, pp. 197–205, 2017.

[20] D. Z. Alhamad and K. Kazi, "Light weight HTTP for transport and content sharing in 5G cellular networks," in *2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy)*, Charlotte, NC, USA, 2014.

[21] S. Wei, V. Swaminathan, and M. Xiao, "Power efficient mobile video streaming using HTTP2 server push," in *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, Xiamen, China, 2015.

[22] J. Qiao, Y. He, and X. S. Shen, "Proactive caching for mobile video streaming in millimeter wave 5G networks," *IEEE*

Transactions on Wireless Communications, vol. 15, no. 10, pp. 7187–7198, 2016.

[23] J. Bruneau-Queyreix, M. Lacaud, D. Negru, J. M. Batalla, and E. Borcoci, "Adding a new dimension to HTTP Adaptive streaming through multiple-source capabilities," *IEEE MultiMedia*, vol. 25, no. 3, pp. 65–78, 2018.

[24] A.'a. Al-Habashna, G. Wainer, and S. Fernandes, "Improving video streaming over cellular networks with DASH-based device-to-device streaming," in *Summer Sim-SPECTS*, Bellevue, Washington, USA, 2017.

[25] J. Qiao, Y. He, and X. S. Shen, "Improving video streaming quality in 5G enabled vehicular networks," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 133–139, 2018.

[26] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming," in *Proceedings of the 2013 workshop on adaptive and learning agents (ALA)*, Saint Paul (Minn.), USA, 2014.

[27] Y. Yang, W. Hu, X. Chen, and G. Cao, "Energy-aware CPU frequency scaling for mobile video streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2536–2548, 2019.

[28] A. Polakovič, R. Vargic, and G. Rozinaj, "Adaptive multimedia content delivery in 5G networks using DASH and saliency information," in *2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP)*, Maribor, Slovenia, 2018.

[29] H. Yousef, J. Le Feuvre, and A. Storelli, "ABR prediction using supervised learning algorithms," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, Tampere, Finland, 2020.

[30] M. S. Asish and R. Aishwarya, "Cyber security at a glance," in *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, vol. 1, pp. 240–245, Chennai, India, 2019.

[31] W. J. Buchanan, S. Helme, and A. Woodward, "Analysis of the adoption of security headers in HTTP," *IET Information Security*, vol. 12, no. 2, pp. 118–126, 2018.

[32] K. Kisa and E. Tatli, "Analysis of http security headers in Turkey," *International Journal of Information Security Science*, vol. 5, no. 4, pp. 96–105, 2016.

[33] F. Y. Osisanwo, J. E. T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

[34] Owasp, *OWASP Top Ten Web Application Security Risks|OWASP|The OWASP Foundation*, https://owasp.org/www-project-top-ten/.

[35] Edgesuite, *Big Buck Bunny*, http://dash.edgesuite.net/akamai/bbb_30fps/.

[36] I. W. Tsang, J. T. Kwok, P.-M. Cheung, and N. Cristianini, "Core vector machines: fast SVM training on very large data sets," *Journal of Machine Learning Research*, vol. 6, no. 4, 2005.

[37] W. Wang, J. Shen, and L. Shao, "Video salient object detection via fully convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 38–49, 2018.