**REDUCING DELAY IN LIVE VIDEO STREAMING**

**OVER 4G NETWORKS USING Q-LEARNING**

**A Project-1 Report**

***Submitted by***

|  |  |
| --- | --- |
| Aakash M | 2016506001 |
| Sriram V | 2016506092 |
| Vishnu M | 2016506110 |

***Under the supervision of***

Dr. Dhananjay Kumar

*In partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**MADRAS INSTITUTE OF TECHNOLOGY CAMPUS**

**ANNA UNIVERSITY, CHENNAI – 600044**

OCTOBER 2020

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this mini-project report titled **“REDUCING DELAY IN LIVE VIDEO STREAMING OVER 4G NETWORKS USING Q-LEARNING”** is the bonafide work of Aakash M (2016506001), Sriram V (2016506092) and Vishnu M (2016506110) who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **Signature** | **Signature** |
| **Dr. M. R. Sumalatha** | **Dr. Dhananjay Kumar** |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| Professor | Professor |
| Department of Information Technology | Department of Information Technology |
| MIT Campus, Anna University  Chennai – 600044 | MIT Campus, Anna University  Chennai – 600044 |

**ACKNOWLEDGEMENT**

It is essential to mention the names of the people, whose guidance and encouragement made us accomplish this project.

We express our thankfulness to our project supervisor **Dr. Dhananjay Kumar**, Professor of the Department, Department of Information Technology, MIT Campus, for providing invaluable support and assistance with encouragement which aided to complete this project.

We are thankful to the panel members **Dr. Radha Senthilkumar,** and D**r. P. Kola Sujatha,** Department of Information Technology, MIT Campus for their invaluable feedback in reviews.

Our sincere thanks to **Dr. M. R. Sumalatha**, Head of the Department of Information Technology, MIT Campus for catering all our needs giving out limitless support throughout the project phase.

We express our gratitude and sincere thanks to our respected Dean of MIT Campus, **Dr. J. Prakash**, for providing excellent computing facilities throughout the project.

**AAKASH M 2016506001**

**SRIRAM V 2016506092**

**VISHNU M 2016506110**

**ABSTRACT**

Live streaming using HTTP Adaptive Streaming (HAS) focuses on delivering appropriate quality of video to the user even during dynamic change in network conditions. The server push segment approach helps in transmission of segments to the client side at a rapid rate. However, this mechanism involves a lot of segments at client buffer, which is a significant encoding overhead as every segment and a larger number of HTTP GET requests is performed to retrieve the segments. In this project, we propose a novel approach of adapting the segment size of the stream dynamically with respect to the buffer occupancy status at the client side based on Q-Learning approach. This approach helps the adaptation engine at client to continuously monitor the packets received and provide feedback to the server based on the condition of the available network to minimise the delay. The implementation of the proposed method deals with segment size adaptation on the top of HTTP/2.0 in a cloud environment, where a 4G wireless network is used a last mile connectivity. The experimental results show that the proposed method provides a significant lower server to display delay during a live video streaming session.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER**  **NO** | **TITLE** | | **PAGE NO** |
|  | **ABSTRACT**  **LIST OF TABLES**  **LIST OF FIGURES**  **LIST OF ABBREVIATIONS** | iv  viii  ix  x | |
| **1** | **INTRODUCTION** |  | |
|  | 1.1 Overview | 1 | |
|  | 1.2 Research Challenges | 1 | |
|  | 1.3 Objective  1.4 Scope of the Project  1.5 Contribution | 2  2  2 | |
|  | 1.6 Adaptive Streaming | 3 | |
|  | 1.7 HTTP/2.0 over HTTP/1.1  1.8 Reinforcement Learning  1.9 Q-Learning  1.10 MPEG-DASH  1.11 H.264 Codec  1.12 NGROK  1.13 Shaka Player  1.14 Organization of the Thesis | 3  4  4  5  6  6  6  7 | |
| **2** | **LITERATURE SURVEY**  2.1 Super Short Segment Approach  2.2 Q-Learning Approach  2.3 Dynamic Segment Size Selection  2.4 Low Latency Live Video Streaming  2.5 Adaptive Video Streaming  2.6 Video Streaming Performance  2.7 Minimize Delay  2.8 Summary of Literature Survey | 8  8  9  9  10  11  11  11 | |
| **3** | **SYSTEM ARCHITECTURE AND DESIGN** |  | |
|  | 3.1 System Architecture | 13 | |
|  | 3.2 Delay Reduction | 14 | |
|  | 3.2.1 Adaptation Mechanism | 14 | |
|  | 3.3 Q-Learning Approach | 15 | |
|  | 3.3.1 Elements of Approach | 15 | |
| **4** | **ALGORITHM DEVELOPMENT AND** **IMPLEMENTATION**  4.1 Machine Learning Based Approach | 17 | |
|  | 4.2 Algorithm  4.3 Selection Policy  4.4 Proposed System Implementation  4.4.1 Server | 17  18  18  18 | |
|  | 4.4.2 Client  4.4.3 State Table  4.4.4 Action Table | 19  19  20 | |
|  | 4.5 Implementation Specification | 20 | |
| **5** | **RESULTS AND DISCUSSIONS** | 21 | |
|  | 5.1 Implementation Environment | 21 | |
|  | 5.2 Server to Display Delay Measurement  5.3 Initial Delay Measurement  5.4 Delay Comparison | 22  23  23 | |
| **6** | **CONCLUSION AND FUTURE WORK** |  | |
|  | 6.1 Conclusion  6.2 Future Work  **REFERENCES** | 24  25  26 | |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TITLE** | **PAGE NO** |
| 4.1  4.2 | State Value Table  Action Value Table  viii | 19  20 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | P**AGE NO** |
| 1.1  3.1  5.1  5.2  5.3  5.4 | Reinforcement Learning  System Architecture  Server to Display Delay  Initial Display Delay  Delay Comparison  Sample Frames | 4  12  22  22  23  23 |
|  |  |  |
|  | ix |  |

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| TCP  RL  MPD  AVC  HAS  MPEG  HTTP  DASH | Transmission Control Protocol  Reinforcement Learning  Media Presentation Description  Advanced Video Coding  HTTP Adaptive Streaming  Moving Picture Experts Group  Hypertext Transfer Protocol  Dynamic Adaptive Streaming over HTTP      x |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Live video streaming is one of the most prevalent method to reach a wider audience over the internet. Providing a near-perfect streaming experience for users without delay is difficult as there are various factors that contribute to the delay. When a video is streamed via online, this problem is more prevalent. Presence of lags and buffering might lead to bad user experience. High delay on streams and in live interactions can have a deep impact on the user experience. Reducing the delay in live streaming platform helps to improve the same. The server to display delay is a primary cause and the codec mechanism used for the purpose of encoding and decoding can also add to a considerable amount of delay. One of the factors that contribute to delay includes this encoding process of the video, but it can be easily eliminated. One of the factor that contribute to the delay is client buffer where the encoded frames are stored. Another factor is the delay at the client end where player downloads the fragments from the server and plays it. The advantage of client buffer is that it reduces the number of interruptions that are caused based on the bandwidth condition thus helping in a smooth play but it also adds up to a few seconds of delay. The underlying network, upon which the video is streamed, can also add to a few seconds of delay.

**1.2 RESEARCH CHALLENGES**

Designing an adaptation engine that involves an algorithm which responds to abrupt bandwidth changes and reduces delay in streaming poses a great challenge in video streaming. Further implementation of the algorithm in real-time 4G wireless network and future 5G wireless network poses major difficulties like bandwidth availability estimation, buffer occupancy status at rapid speed, number of segments and adaptive pushing of segments. Defining the algorithm that would reduce the delay in streaming between a single server and multiple receivers also poses a great challenge.

**1.3 OBJECTIVE**

The major objective is to reduce the end-to-end delay in live streaming of a video by implementing a segment size adaptation algorithm based on Q-learning approach. The Q-learning algorithm at the client’s side needs to be optimized which analyses the current status of buffer, and report server over an HTTP/2.0 connection for adaptation.

**1.4 SCOPE OF THE PROJECT**

The proposed system design focuses on reduction of the camera-to-display delay/latency and end-to-end delay in a live video streaming thus enabling the continuous streaming of video on the user mobile with minimum latency. The streaming system uses reinforcement learning technique on the top of HTTP connected through 4G wireless networks. The FFMPEG is to be used for the video scaling and transcoding as per H.264 video coding standard.

**1.5 CONTRIBUTION**

The project contributes to reducing the delay of live video stream by developing an adaptive engine at client which ensures the continous flow of live video stream in variable bandwidth conditions. A general approach of MPEG-DASH results in a considerable amount of delay, whereas the implementation of Q-Learning in the system helps to further minimize delay in real time. The system was tested between two laptops connected over 4G Hotspot in a real time scenario.

**1.6 ADAPTIVE STREAMING**

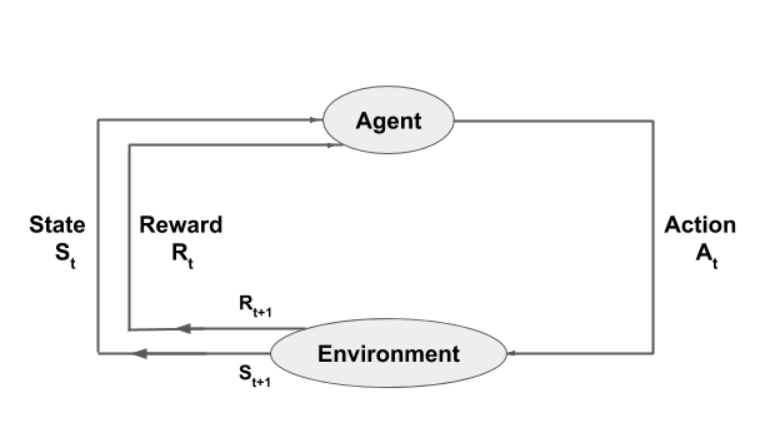
Adaptive Streaming is the mechanism of encoded streams into many different bitrates and breaking them into fragments. These multi-bitrate chunks would then be indexed in a manifest file and delivered to the player. This file consists of video and audio content additionally to different information like program data, subtitles or different metadata. This information may well be separated into peculiar fragment files or would possibly get combined within the file fragment. The fragments are delivered to the client, hosted on a HTTP server. The various sequences of fragments that differ in bitrate and codecs may represent a similar content. The client requests the fragments by playing the stream from HTTP server. As the fragments are downloaded the client plays the fragment unceasingly. The video playback is made by downloading all the video, audio and different information that are held in separate fragment sequences. Audio quality also adjusts on available bandwidths and each fragment is encoded separately. H264 encoding technique is the primary technique used. Thus it removes the dependency on the previous frames and progressive stream and adapts to the current bandwidth and produces optimal streams.

**1.7 HTTP/2.0 OVER HTTP/1.1**

HTTP 2.0 uses a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. This advanced feature to download web files asynchronously from one server. It also allows a server to push content to a client before the client asks for it. The server also sends a message letting the client know about the content it is above to receive. This results in reducing the round trip time for setting up numerous TCP connections and in turn helps in improving the speed of the website. It also removes the need of domain sharding. HTTP 2.0 also supports an additional feature called prioritization, enabling requests to be sent out contingent upon their dimension of significance to a specific website page. HPACK header compressions diminishes the overload of redundant information in each header packet.

**1.8 REINFORCEMENT LEARNING**

Reinforcement Learning (RL) is the training of a machine learning model that helps agent to arrive at a goal. It enables the model to get feedback from its very own activities and encounters. Despite the fact that both supervised and reinforcement learning use mapping among input and output, unlike unsupervised learning, RL uses rewards and penalties as cues for right and wrong behaviours and the goal is to maximize the reward. It helps the model with a clue in order to solve a problem. It enables the model to start from random trials and learning over the course and reaching a maximum reward. It produces an optimized result. Figure 1.1 speaks to the components associated with a reinforcement learning model.



**Figure 1.1** - Reinforcement learning

**1.9 Q-LEARNING**

[**Q-Learning**](https://en.wikipedia.org/wiki/Q-learning) is a commonly used model free approach. It revolves around the notion of updating Q values which denotes value of doing action ‘a’ in state‘s’. The value update rule is the core of the Q-learning algorithm.

**(1.1)**

where ‘s’ is the current state, ‘a’ is the current action, ‘rwd’ is the associated reward, ‘s’’ is the new state after action ‘a’, ‘a’’ is the new action from the state ‘s’’ , the learning rate ‘α’ indicates how much the acquired information will affect the old value of Q(.) in updating to new value and the discount factor ‘γ’ that weighs the contribution of the immediate and future rewards (0≤ γ≤1).

**1.20 MPEG-DASH**

The basic plan of MPEG-DASH is to cut the media file into segments which might be encoded at completely different bitrates or spatial resolutions. These segments may be downloaded through HTTP standard compliant GET requests, where the HTTP Server serves three completely different qualities, i.e., Low, Medium and best sliced into segments of equal length. The adaptation to the bitrate or resolution is carried out on the client side for every individual segment, e.g., the client will switch to the next bitrate according to the bandwidth availability. This has many pros since the client is aware of received throughput and the context of the user best.

In order to explain the temporal and structural relationships between segments, MPEG-DASH introduced the supposed Media Presentation Description (MPD). The MPD is an XML file that represents the various qualities of the media content and the individual segments of every quality with HTTP Uniform Resource Locators (URLs). This structure provides the binding of the segments to the bitrate, resolution, start time, period of segments, etc. As a consequence, every client first request the MPD that contains the temporal and structural information for the media content and supported to that information it'll request the individual segments that match best for its needs.

**1.21 H.264 CODEC**

H.264 or MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC) is a block-oriented motion-compensation-based video compression standard. It's the foremost format used for recording, compression, and distribution of video content. It supports resolutions up to 8192×4320, together with 8K UHD. The intent of the H.264/AVC project was to make a standard capable of providing good video quality at considerably lower bit rates than previous standards, while not increasing the design complexity most that it'd be impractical or too expensive to implement. The H.264 standard is often viewed as a "family of standards" composed of various profiles. A selected decoder decodes a minimum of one, however not essentially all profiles. H.264 is usually used for lossy compression, though it's attainable to make actually lossless-coded regions inside lossy-coded photos or to support rare use cases that the entire encoding is lossless.

**1.22 NGROK**

NGROK is a web server that exposes local servers on a machine to public internet over tunnels. It is a cloud based server that accepts input from a local host and publishes it on a public address for global access. It creates a secure HTTPS connection for the program running on the local developmental machine and sets authenticated access to the same. It is powered using HTTP/2 mechanism and hence it bolsters the loading speed of the tunnel. It also supports a multiple integration environment for multiple tunnels and help in accessing them through the help of a web socket.

**1.23 SHAKA PLAYER**

Shaka Player is a Javascript based framework used in playing adaptive media streams. It helps to play Dynamic Adaptive Streaming Content by compiling the stream information from the host address and embedding it at the client end and playing the media adaptively using the conditions specified by the user. It works with the help of Media Source Extension and can be played in a browser.

**1.24 ORGANIZATION OF THE THESIS**

The rest of the thesis is organized as follows. Chapter 2 presents the literature survey on video streaming on HTTP/2.0 protocol and delay reduction by various approaches. Chapter 3 models the architecture and system design, outlining the architecture and design of the proposed system. Chapter 4 explains about the implementation details, explaining the specifications and environment. The results achieved are presented in Chapter 5. Chapter 6 presents the conclusion and some possible avenues for future research on the topic.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 SUPER-SHORT SEGMENT APPROACH**

Super Short Segments (SSS) approach deals about dividing the video content into different segments size ranging from 1-10 seconds, each encoded at multiple quality levels (Van Der Hooft J. et al., 2018). The end-to-end delay in HTTP Adaptive Streaming (HAS) is because of frequent variations in the network conditions. The authors have made use of the HTTP/2’s push feature to actively push segments from server to the client. The main input parameters used are the bandwidth of the network and the video player’s buffer filling level. The proposed approach showed to reduce start up delay by 31.2% when compared to other solutions over HTTP/1.1 and reduce end-to-end delay by 4 seconds to provide the user a continuous play-out experience.

**2.2 Q-LEARNING APPROACH**

Q-learning is a value based reinforcement learning approach that is based on Off-Policy TD learning. The value-based algorithm also uses a policy based estimate algorithm for choosing the value function which results in a better final performance (Bekcheol Jang et al., 2018). Analysis showed that Q-learning is slow in learning and but performs in the local minimum itself and The Q-learning algorithm directly tunes the Q-values, and then the Q-values will indirectly affect the action selection policy. The comparison of experimental results proved that the proposed method performs better than other reinforcement learning algorithms taken overcoming all the disadvantages.

**2.3 DYNAMIC SEGMENT SIZE SELECTION**

Dynamic segment size can be used to reduce the quality of the video streaming whenever network condition are not favourable so that the user has a continuous stream of the video (Bedogni L. et al., 2017). When the network conditions are good, the video is streamed to the user using a larger segment size. The continuous monitoring of the network conditions and the segment size is decided by the MPEG-DASH client with the input parameters as latency, mobility of the client, the data rate of the connection and the throughput of the network. Using this technique, the system successfully provided a smoother playback and increased the quality of the stream for the client.

A DSDC scheme (Dynamic Segment Duration Scheme) can be used to reduce buffering delay and provide seamless playback for the user (Yun D. et al., 2016). Buffering delay is an important parameter for seamless playback. The segment size is determined based on input parameters as client buffer occupancy and segment fetch time variation, i.e., load latency. The results were simulated and DSDC scheme proved to provide seamless playback and maintain low buffering delay. Ensuring a seamless playback has proved to be more advantageous on the grounds of Quality of Experience.

**2.4 LOW LATENCY LIVE VIDEO STREAMING**

More latency occurs in video streaming with the current adaptation schemes techniques. A mathematical model for adaptive live streaming can be used to reduce the latency (Shuai Y. et al., 2018). The main parameters to determine delay used are the network throughput and video bitrate. Depending on the network parameters which are throughput and delay, the required buffering delay is presented for a smooth continuous stream. Finally the approximation is validated under a trace driven simulation of a given network in video streaming.

A low latency live video streaming can be achieved using HTTP chunked encoding technique (Swaminathan V. et al., 2011). The analysis showed that the existing HTTP streaming solutions cannot provide a low latency experience due to the fact that in all of the existing methodologies, delay is calculated based on the duration of the media fragments which are individually requested over HTTP. The authors have proposed a low latency HTTP streaming approach using HTTP chunked encoding, which enables the server to transmit partial fragments before the entire video fragment is published. The analysis and experimental results showed that the chunked encoding approach is capable of reducing the live latency to one to two chunk durations and that the resulting live latency is independent of the fragment duration.

The new HTTP/2 was standardized in 2015. This technique uses the HTTP/2 features such as server push and stream termination to shorten camera to display delay. The results proved that with small buffer size users experience high video bit rate and stable buffer level. Also buffer size is reduced by 2 seconds without sacrificing video quality.

**2.5 ADAPTIVE VIDEO STREAMING**

Video Streaming must be seamless without latency over the network. When the network conditions are good, the video is streamed to the user using a larger segment size with respect to the size of the buffer (Zhang X. et al., 2018). A fast internet connection has a higher bitrate than a slow internet connection. The internet by its very nature is an incredibly complex network of connections and systems and the performance of these systems are constantly changing.. MPEG DASH based streaming with the consideration of parameters and metrics will help in providing playback that results in minimising the delay in the live stream.

**2.6 VIDEO STREAMING PERFORMANCE**

Video streaming over a wireless network is difficult because of the dynamic radio propagation environment, limited radio resources and Quality of Service must be available at the minimum acceptance level for a smooth continuous stream (Mohd Ramli H. et al., 2011). The existing proposed systems for reducing latency are computationally expensive to be used in Orthogonal Frequency Division Multiple Access based wireless IP networks. The evaluation of the performance of the video streaming was done under three famous algorithms. The demonstrations proved that Propositional Fair algorithm performs better than the other well-known algorithms by providing minimum acceptance level.

**2.7 MINIMIZE DELAY**

Having a high latency on streams, and especially in interactive live experiences, can have a significant impact on user experience. A user’s network connection can have a significant impact on latency (Shuai Y. et al., 2018). A similar trade-off appears when looking at quality of video compared to its latency. Achieving a higher quality for the end-user often results in higher bandwidth requirements due to higher resolutions and frame rates, or more time needed on the encoder side in order to shrink the bitstream at high quality. Switching latency between adaptive bitrate streams are a suitable way to tackle this and this is used in GOP.

**2.8 SUMMARY OF THE LITERATURE SURVEY**

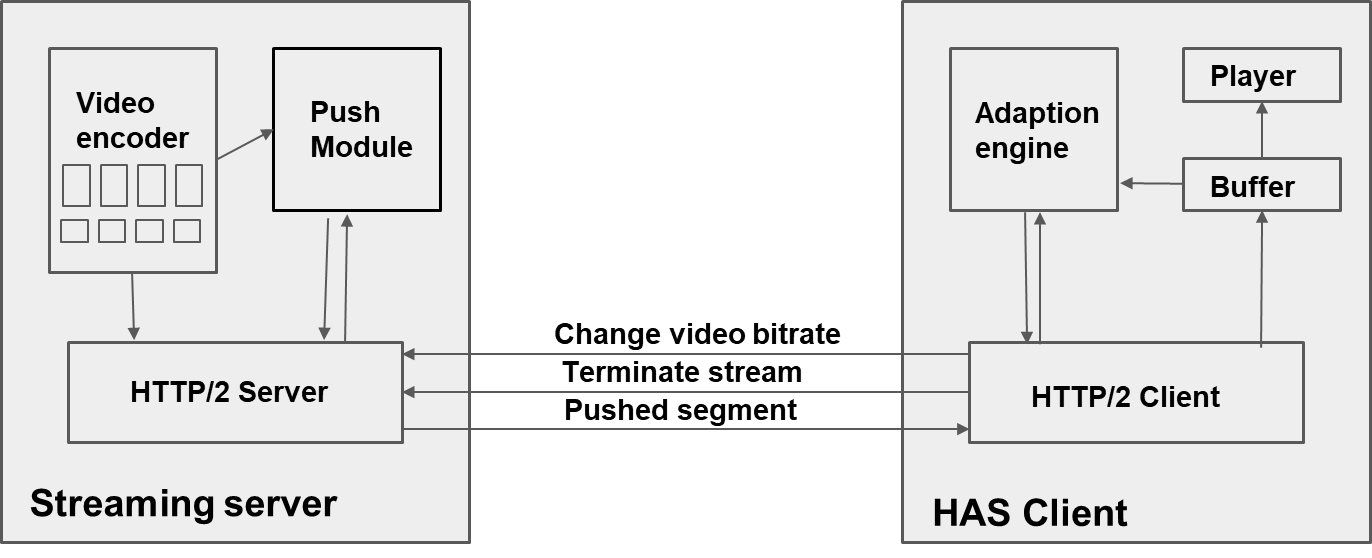
Approach of super short segments to reduce delay works along with the learning process of the client based on the analysis of the flow of stream in a fluctuating bandwidth conditions. In the dynamic segment selection feature based on the learning of adaptation engine, reducing further delay remains a challenge. Low latency live stream and the parameters involved using an adaptive streaming mechanism based on MPEG-DASH was analysed by many along with the performance of the video stream and approaches to minimize the delay in varying network environment but the application of reinforcement learning in selecting the segment sizes in dynamic bandwidth conditions could be a better idea for improvement. The Q-Learning approach at adaptation engine with dynamic segment size selection to minimise delay needs further investigation.

**CHAPTER 3**

**SYSTEM ARCHITECTURE AND DESIGN**

1. **SYSTEM ARCHITECTURE**

The client side of the architecture holds the adaptation engine that measures the available buffer occupancy status. The client on a regular time interval runs the adaptation engine to determine the suitable segment size so that the buffer neither overflows or underflows. Heuristic algorithms that perform bit-rate adaptation are not closer to the optimal solution and also they are very complex to implement. Hence, Reinforcement learning proves to be an efficient solution. In reinforcement learning, the learning agents are permitted to try different state-action combinations and they steadily learn the best strategy for every situation. The agent employs trial and error to come up with a solution to the problem. Hence we use the Q-Learning approach, a reinforcement learning algorithm that depends on off-value Temporal Difference policy. The server encodes the frames using H.264 video codec and pushes the same to the client and the streaming starts. As shown in Figure 3.1, the client requests the Media Description File from the server and receives it and the adaptation engine present performs the analysis process on the available buffer status and decides the parameters to be monitored. The server receives the feedback from the client and acts upon it and sends the frames of the appropriate size in order to reduce the delay and ensure smooth streaming at the client end. It trains the client’s adaptation engine and determines the best action to be taken by the server in future for obtaining better results. The architecture diagram of the overall proposed system is shown in Fig. 3.1. The diagram depicts that the server streams the video to the client using HTTP/2.0 protocol with TCP as the underlying transport layer protocol.



**Figure 3.1** – System Architecture

**3.2 DELAY REDUCTION**

The playback in video streaming is a primary factor for the users and the presence of a huge delay in the stream will lead to a bad user experience. Hence to reduce the delay efficiently, the client must be monitored so that it does not fall below a certain threshold value of buffer underflow. To avoid this, the segment size at which the server sends the video chunks to client must be altered based on the buffer occupancy status at the client side.

**3.2.1 ADAPTATION MECHANISM**

The client requests for a live video stream from the server. The server initially responds with the Media Description File and with video chunks of least segment size. The client monitors the buffer occupancy level at regular intervals and conveys the detail to the adaptation engine. The adaptation engine based on the client’s buffer occupancy status, determines the state in the Q-matrix of the Q-learning approach, by a mere mapping technique. The adaptation engine determines the suitable action from the Q-matrix by comparing all the Q-values for the state and sorting out the largest Q-value. This action is responded to the server, and the server responds with video chunks of new segment size. With downloading chunks at new segment size, the buffer occupancy level at the client side, now reaches a new value and thus it reaches a new state, according, to the Q-matrix. For this new Q-value, the new optimal action is determined. With these current states and new states, the reward is calculated for the current state, i.e., the value based on quality estimation is used as reward and based on the reward obtained the Q-matrix is updated.

**3.3 Q-LEARNING APPROACH**

The Q-Learning Approach is a standard approach. All the Q values are determined for all the state action pairs alike in the Q-learning approach. Along with the Q-values, the rewards obtained and the new state reached, these details are also stored. When the goal state in the current episode is reached by the agent, the stored values are utilized for updation of Q-values.

**3.3.1 Elements of Q-Learning Approach**

The important elements of Q-Learning approach are listed as follows:

*a) State*

State contains the date about the environment conditions at each time. In this approach, the current state is characterized as the buffer occupancy values estimated in percentage.

*b) Action*

The new segment size with which the chunks to be delivered to the client by the server is considered as action.

*c) Reward Function*

The function that shows, how optimal is the action chosen, is the reward function.

*c) Q-Table Q(S, A):*

The rows of the matrix are all the states of the system and each column contains one the possible actions, i.e., segment sizes. For a given pair (s, a), Q (.) indicates the learned benefit that the system will get taking action ‘a’ in state‘s’. In order to include what the client has learned taking an action, the Q-Learning approach updates this matrix after each quality decision as follows

**(3.1)**

where ‘s’ is the current state, ‘a’ is the current action, ‘rwd’ is the associated reward, ‘s’’ is the new state after action ‘a’, ‘a’’ is the new action from the state ‘s’ , the learning rate ‘α’ indicates how much the acquired information will affect the old value of Q(.) in updating to new value and the discount factor ‘γ’ that weighs the contribution of the immediate and future rewards (0≤ γ≤1).

**CHAPTER 4**

**ALGORITHM DEVELOPMENT AND IMPLEMENTATION**

**4.1 MACHINE LEARNING BASED APPROACH**

Machine Learning based approach is used in improving the reliability and performance of live stream by reducing the delay. Since the implementation is done in real time, both Supervised Learning and Semi-supervised Learning cannot be used since they don’t interact with the environment. Hence Reinforcement learning is used which helps in finding an optimal approach in a real time scenario. Further, Q-Learning approach which is based on Off-Policy Learning is implemented such that it helps in improving the action taken based on a reward in the current condition to minimise the delay. To select the best value from the Q-Table, Selection Policy is used.

**4.2 ALGORITHM**

1. Start

2. Initialize the Q-matrix consisting of state (s) and action (a) values, learning rate (α) and discount factor (γ)

3. Repeat for each state:

3.1 Observe current state (s) based on buffer availability.

3.2 Perform action (a) for the current state using selection policy.

//Selection policy function below

3.3 Compute the reward *(rwd)* value.

3.4 Update the value of reward (*rwd*) with the calculated value.

3.5 Compute new state (s') by measuring the buffer availability status.

3.6 Perform new action (a') and update the value of (s') using selection

policy.

3.7 Update the newly computed values in the Q-Table.

𝑄’(𝑠,𝑎)→𝑄(𝑠,𝑎) + 𝛼.[ *rwd* + 𝛾.𝑚𝑎𝑥 𝑄(𝑠’,𝑎’) − 𝑄(𝑠,𝑎)]

3.8 Update s' value (s 🡨 s')

3.9 Update new action (a 🡨 a')

3.10 The newly updated s and a values are sent to server.

4. End

**4.3 SELECTION POLICY**

1. Start

2. Initialize max <- 0, action <- 0, flag <- 0

3. For a = 1 to n:

3.1 Initialize Q(s,a)

4. For i=1 to n:

4.1 if Q(s,a) >= max:

4.1.1 max 🡨 Q(s,a)

4.1.2 action 🡨 i

5. End

**4.4 PROPOSED SYSTEM IMPLEMENTATION**

The system architecture proposed encompasses a client-server model. A schematic diagram of the client-server model along with its main functions. It represents the individual functions of the client and server and how their functionalities connect each other.

**4.4.1 Server**

The media locator URL of the server is initially set either to d-show API, that accesses the camera to capture the live video from the host system. The feed from the camera or from the stored video is encoded. Segments and Manifest Files are created accordingly.

**4.4.2 Client**

At the client side, the Media Player Description file plays an important role. The Media Player Description (MPD) file created by the server has the details of the segments regarding the live feed. The client gets the information about the stream such as the bitrate, frame rate of the stream and also the segment details. The media player is modified to handle streaming data. As the MPD file gets updated from time to time, the player should check the contents from the source buffer.

**4.4.3 State**

The buffer occupancy values are mapped to the state and each current state is shown in the Table 4.1.

**Table 4.1** – State Table

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Buffer Occupancy Value (In percentage)** | **State** |
| 1 | 0 – 10 | 1 |
| 2 | 10 – 20 | 2 |
| 3 | 20 – 30 | 3 |
| 4 | 30 – 40 | 4 |
| 5 | 40 – 50 | 5 |
| 6 | 50 – 60 | 6 |
| 7 | 60 – 70 | 7 |
| 8 | 70 – 80 | 8 |
| 9 | 80 – 90 | 9 |
| 10 | 90 – 100 | 10 |

**4.4.4 Action**

The action values mapped to the segment sizes accordingly and are shown in the Table 4.2.

**Table 4.2** – Action Table

|  |  |
| --- | --- |
| **Action** | **Segment Size (in seconds)** |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |

**4.5 IMPLEMENTATION ENVIRONMENT**

JavaScript programming environment is used for implementation purpose, since it runs on all browsers and is platform independent. For playing the media, Shaka player is used. Shaka is an open-source DASH player from Google. The server and the client both support MPEG – DASH. The client and server were implemented in Windows 10 (64 bit operating system). The client and server were connected to the 4G Mobile Hotspot network. The streaming was implemented on top of the HTTP/2.0 protocol with TCP as its underlying transport layer protocol.

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

**5.1 IMPLEMENTATION ENVIRONMENT**

A client-server environment for the dynamic adaptive streaming over HTTP is created. The client requests the server to send the video after getting connected via HTTP port. The details of the live feed get stored in the media player description file and the file is passed to the client. At the client side, the segments which are sent by the server are appended to the buffer and the player plays the contents from the buffer.

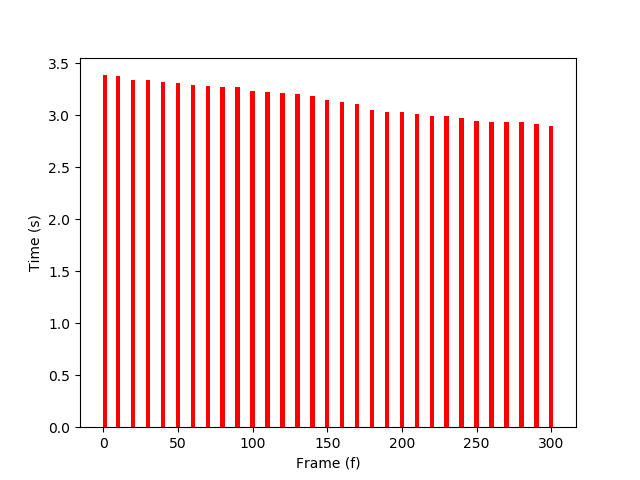
The proposed system is implemented and tested using an ACT broadband connection at the server end and an Airtel 4G Hotspot at the client end. The connection is established using TCP and it is supported on top of HTTP/2.0. The server uses NGROK tool along with d-show API to establish a connection with the client. The client receives the frames using Shaka Player and plays it for display.

**5.2 SERVER TO DISPLAY DELAY MEASUREMENT**

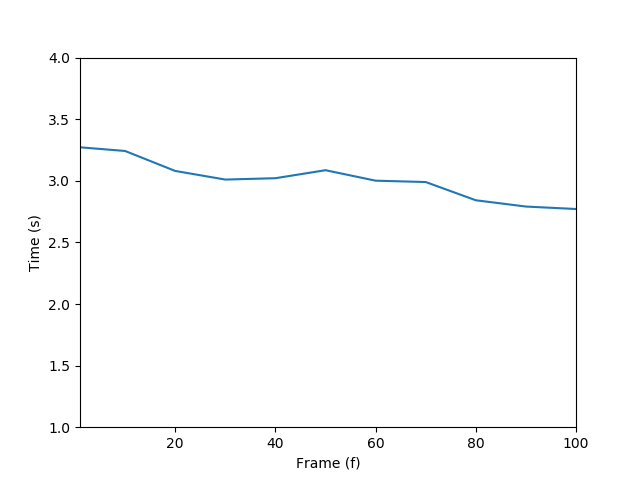
The Server to Display delay is a metric that determines the latency between a frame getting generated at the server side, pushed to stream and the frame viewed by the user at the client side. The Server to Display delay is calculated during the live streaming experiments and is plotted in Figure 5.1.

**5.3 INITIAL DELAY MEASUREMENT**

The initial server to display delay depends upon the segment size with which the live stream is initiated. The delay is a cumulative value of the segment size in seconds and the segment propagation time at the network to the client. The initial server to display delay for various initial segment size is shown in Figure 5.2.



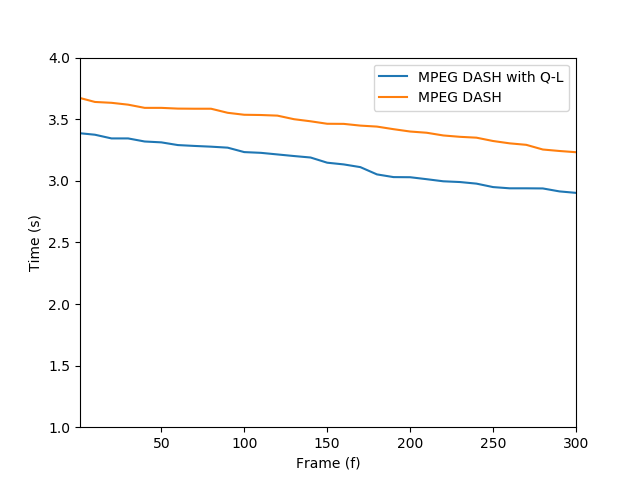
**Figure 5.1** – Server to Display Delay



**Figure 5.2** – Initial Display Delay

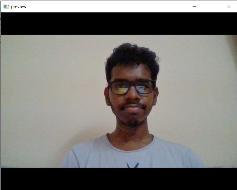
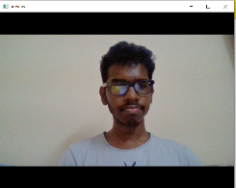
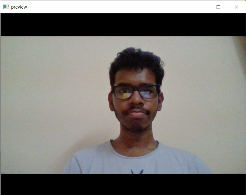
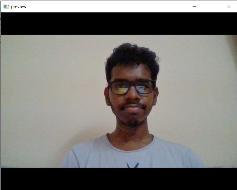
**5.4 DELAY COMPARISON**

The delay for the transfer of live stream was computed using a general MPEG-DASH approach and also with the implementation of Q-Learning. It shows that Q-Learning helps in reducing the delay further compared to a generic approach of DASH. This applies to two factors: the adaptive size of segments and buffer filling at client end. The observed delay for the two different approaches is plotted in Figure 5.3.



**Figure 5.3** – Delay Comparison

Few frame sequences of original video from the server side while experimenting with live streaming is depicted in Figure 5.4.



**Figure 5.4 –** Some frames captured during live streaming.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 CONCLUSION**

A live video streaming was implemented in a 4G wireless network on top of HTTP/2.0. The various challenges faced due to the changing link conditions are studied. Reduction of the delay is a requirement for improving the quality of the video and the overall quality of experience. As HTTP/2.0 is the protocol used here, the number of round trips are reduced and the delay reduction depends on the segment size of the video chunk which is pushed upon the request from client. The proposed system identifies the current buffer occupancy status at the client side. On a sudden bandwidth drop, or an abrupt change in the network, the buffer occupancy status gets greatly varied. The video chunks are sent in accordance with the status of the buffer at the client.

The buffer at client end may also get segments from the server very quickly that can lead to rapid pooling of segments at client side or slow down the segment arrival rate due to network congestions that could lead to the draining of available segments at the client side. The adaptation engine in the client side, determines the segment size with which the server must stream the chunks of video data to the client at regular intervals, so that there is neither any freezing occurring while playing the video nor the buffer faces any overflow or underflow. The Q-Learning works for streaming applications, but to achieve a better quality of experience for interactive applications, the streaming latency needs to be further reduced. The proposed method with experimental results could help in standardizing the future live streaming services and in delivering a minimum delay setup.

**6.2 FUTURE WORK**

The proposed system was implemented in one way video communication, but it can be extended to support two-way streaming. The system can also be structured in such a way that it enables the server to serve more than one client at a time. Furthermore, the streaming system need to provide adaptive support to both audio and video data streaming simultaneously at a very low latency. The system level implementation could be based on the ITU-T recommendation P.1203.1 which defines the MOS computation in a standard procedure.

**REFERENCES**

1. Bedogni, L., Di Felice, M. and Bononi, L., (2017), “Dynamic segment size selection in HTTP based adaptive video streaming”. In proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2nd May’ 17, pp. 665-670.

2. Bekcheol Jang, Myeonghwi kim, Gaspard Harerimana (2018), “Q-Learning Algorithms: A Comprehensive Classification and Applications”, In proceedings of the 2018 IEEE Conference on Computer Communication, 23rd October’ 18, pp. 294-296.

3. Huang, T.Y., Johari, R., McKeown, N., Trunnell, M. and Watson, M. (2015), “A buffer-based approach to rate adaptation: Evidence from a large video streaming service”, ACM SIGCOMM Computer Communication Review, 44(4), pp.187-198.

4. Huysegems, R., Van Der Hooft, J., Bostoen, T., Rondao Alface, P., Petrangeli, S., Wauters, T. and De Turck, F. (2015), “HTTP/2-based methods to improve the live experience of adaptive streaming”. In Proceedings of the 23rd ACM international conference on Multimedia, 28th October’ 15, pp. 541-550.

5. Hooft J.V.D., S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F.D. Turck (2017), “An HTTP/2 push-based approach for low-latency live streaming with super-short segments”, Journal of Network and Systems Management, March’ 17, vol. 26, no. 1, pp. 51-78.

6. Le, H.T., Nguyen, T., Ngoc, N.P., Pham, A.T. and Thang, T.C., (2018), HTTP/2 Push-Based Low-Delay Live Streaming Over Mobile Networks With Stream Termination”, IEEE Transactions on Circuits and Systems for Video Technology, 28(9), pp.2423-2427.

7. Lohmar, T., Einarsson, T., Fröjdh, P., Gabin, F. and Kampmann, M., (2011), “Dynamic adaptive HTTP streaming of live content”. In proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, June’ 11, pp. 1-8.

8. Mohd Ramli, H., Sandrasegaran, K., Basukala, R., Patachaianand, R. and Afrin, S., (2011), “Video Streaming Performance Under Well-Known Packet Scheduling Algorithms”, International Journal of Wirless and Mobile Networks.

9. Nguyen, D.V., Le, H.T., Nam, P.N., Pham, A.T. and Thang, T.C., (2016), “Adaptation method for video streaming over HTTP/2”, IEICE communications Express, 5(3), pp.69-73.

10. Shuai, Y. and Herfet, T., (2018), “Towards reduced latency in adaptive live streaming”. In proceedings of the 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), 13th January’ 18, pp. 1-4.

11. Sobhani, A., Yassine, A. and Shirmohammadi, S., (2017), “A video bitrate adaptation and prediction mechanism for HTTP adaptive streaming”, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 13(2), p.18.

12. Stockhammer, T., (2011), “Dynamic adaptive streaming over HTTP: standards and design principles”. In Proceedings of the second annual ACM conference on Multimedia systems, 13th February’ 11, pp. 133-144.

13. Swaminathan, V. and Wei, S., (2011), “Low latency live video streaming using HTTP chunked encoding”. In proceedings of IEEE 13th International Workshop on Multimedia Signal Processing, 18th October’ 11, pp. 1-6.

14. Thang, T.C., Le, H.T., Pham, A.T. and Ro, Y.M., (2014), “An evaluation of bitrate adaptation methods for HTTP live streaming”, IEEE Journal on Selected Areas in Communications, 32(4), pp.693-705.

15. Ur Rahman, W. and Chung, K., (2015), “Buffer-based adaptive bitrate algorithm for streaming over HTTP”, KSII Transactions on Internet and Information Systems (TIIS), 9(11), pp.4585-4603.

16. Van Der Hooft, J., Petrangeli, S., Wauters, T., Huysegems, R., Bostoen, T. and De Turck, F., (2018), “An HTTP/2 push-based approach for low-latency live streaming with super-short segments”, Journal of Network and Systems Management, 26(1), pp.51-78.

17. Wang, Y.H., Li, T.H.S. and Lin, C.J., (2013), “Backward Q-learning: The combination of Sarsa algorithm and Q-learning”, Engineering Applications of Artificial Intelligence, 26(9), pp.2184-2193.

18. Wei, S. and Swaminathan, V., (2014), “Cost effective video streaming using server push over HTTP 2.0”. In proceedings of IEEE 16th International Workshop on Multimedia Signal Processing (MMSP), 16th September’ 14, pp. 1-5.

19. Xu, Z., Zhang, X. and Guo, Z., (2018), “QoE-driven Adaptive K-Push for HTTP/2 Live Streaming”, IEEE Transactions on Circuits and Systems for Video Technology.

20. Yun, D. and Chung, K., (2016), “Dynamic segment duration control for live streaming over HTTP”. In proceedings of the International Conference on Information Networking (ICOIN), 14th January’ 16, pp. 206-210.

21. FFmpeg - <https://www.FFmpeg.org>

22. NGROK - Secure introspectable tunnels to localhost <https://ngrok.com/>

23. NGINX – Web server for web serving [https://www.nginx.com/](https://www.nginx.com/%20)

24. Speedtest by Ookla - <https://www.speedtest.net/>