國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於長短期記憶模型之智慧救護車自適應比特率影像直播系統

EMS-RTC LSTM-based Adaptive Video Streaming for Smart Ambulance

王柏文

Bo-Wen Wang

指導教授: 蔡欣穆 博士

Advisor: Hsin-Mu Tsai Ph.D.

中華民國 112 年 6 月

June, 2023

doi:10.6342/NTU202301100

致謝

感謝沈雯萱學姐，在我碩士班的時候花很多時間跟我 meeting，討論東西要怎麼弄，包括最後的口試投影片以及口試論文還麻煩學姐幫忙看，真的很感謝學姐

感謝指導教授蔡欣穆老師這些年來的指導跟研究上給予的建議，終於順利畢業了，謝謝老師

doi:10.6342/NTU202301100

摘要

一台智慧型救護車會不斷地分享病人的生命徵象數據和救護車內的影像流，讓駐院醫生可以與救護人員合作，對病人進行早期評估甚至提供治療指南。不過要讓醫生和救護人員之間合作順暢的前提是要有高品質的影像串流。影像品質差或延遲高的串流會大幅增加對病人的準確診斷的困難度。

由於需要從移動的救護車傳送直播視訊，因此串流的效能受到行動網路的頻寬快速變化的影響。在這種情況下，影片串流的比特率能夠快速適應行動網路服務品質的變化並相應地進行調整是至關重要的。若影片串流的比特率超出行動網路的可用頻寬，就有可能會發生卡頓以及破圖。我們必須讓醫生可以透過直播來判斷病人的病況，要做到這件事，直播的時候必須儘量避免卡頓和破圖的發生。為了達到這件事，我們必須使所用頻寬不超過頻道的可用頻寬。

為此，我們提出了 EMS-RTC，一個針對救護車服務需求專門設計的即時視頻串流平台。EMS-RTC 訓練了一個分類器，根據在現實駕駛情況下收集的一系列信號和網絡相關指標的特徵，推斷出最佳的比特率。我們將 EMS-RTC 與目前最具代表性的自適應串流演算法（即 Google 擁塞控制，GCC）進行比較。從真實世界的行駛情況中收集的評估結果顯示，EMS-RTC 可以在可忽略降低一點影像品質的成本下，將卡頓事件的總時間長度減少 3.85 倍。

doi:10.6342/NTU202301100

關鍵字：長短期記憶模型、自適性比特率、使用者體驗品質、Google 擁塞控制

doi:10.6342/NTU202301100

Abstract

A smart ambulance continuously shares the patient's vital signs data and video feeds from the ambulance so that the doctors stationed in the hospital can work with the paramedics in the ambulance, performing an early assessment of the patient or even providing treatment guidelines. One prerequisite to the seamless cooperation between the doctor and the paramedics is high-quality video streaming. Video streaming with bad image quality or high latency would introduce significant difficulty in providing accurate diagnoses of the patient. As the video needs to be delivered from a moving ambulance, the streaming performance is influenced by the fast variation of the service quality of the mobile network. Compared to static scenarios, additional channel impairments, such as the shadowing effect, is introduced in the mobile scenarios. In this case, it is crucial that the bitrate of the video streaming can quickly react to the changes in the service quality and is adjusted accordingly, such that the video quality leverages full available bandwidth, but does not exceed what the channel can support. To this end, we propose EMS-RTC, a real-time video

doi:10.6342/NTU202301100

streaming platform specialized for the need for ambulance services. EMS-RTC trains a classifier to infer the optimal bitrate based on features from a range of signal- and network-related metrics, collected in real-world driving scenarios. We compare EMS-RTC to a state-of-the-art ABR algorithm (i.e., Google congestion control, GCC). Evaluation results collected from real-world driving scenarios show that EMS-RTC reduces the total time duration of stall events by a factor of 3.85, at the cost of a neglectable reduction of image quality.

Keywords: Long Short-Term Memory, Adaptive Bitrate, Quality of Experience, Google Congestion Control

doi:10.6342/NTU202301100

Contents

doi:10.6342/NTU202301100

doi:10.6342/NTU202301100

List of Figures

doi:10.6342/NTU202301100

doi:10.6342/NTU202301100

List of Tables

doi:10.6342/NTU202301100

Chapter 1 Introduction

1.1 Introduction

Ambulance services, also known as emergency medical services (EMSs), deliver initial care for patients with emergency health problems. In ambulances, paramedics and emergency medical technicians (EMTs) constantly work under pressure to provide initial diagnosis and proper treatment to the patients to save lives. However, on the emergency scene, they sometimes require opinions or guidance to treat the patient correctly and timely since they might have limited experience with issues pertaining to the patient. For example, agonal breathing is a common symptom in patients with out-of-hospital cardiac arrest (OHCA). Immediate recognition of agonal breathing and early CPR is critical for the survival of patients in emergencies. However, despite education on how to determine agonal respiration, many Emergency Medical Technicians (EMTs) struggle to identify it in practice, leading to delays in treatment and potentially fatal outcomes for patients. In order to improve the survival rates of patients, doctors aim to monitor the patient's physical condition and provide real-time guidance to EMTs in the ambulance. To achieve this goal, the development of a smart ambulance is necessary.

With the advance in communication technologies, smart ambulances could connect the ambulance with a hospital or a dispatch center using 4G/5G networks [1]. As illustrated

doi:10.6342/NTU202301100

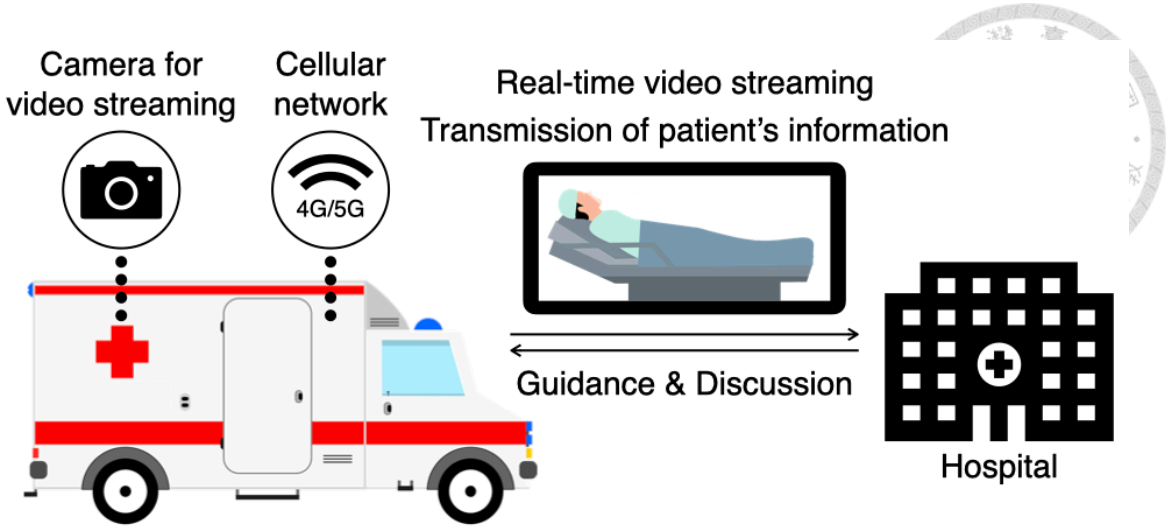


Figure 1.1: Illustration of a smart ambulance.

in Fig 1.1, smart ambulances are equipped with video streaming devices and cameras to deliver the videos of the patients to the hospital, allowing real-time remote consultation and discussion with doctors while the patients are in transit. Besides, necessary treatment can be administered to the patient under the guidance of doctors. This can be crucial as the experienced doctors at the hospital can advise making early preparations to expedite critical treatments after the patient arrives at the emergency department. Shortening the time to administer such treatments is often vital to increase the survival rate of the patient.

To ensure seamless collaboration between doctors and on-site paramedics, the quality of video streaming plays an important role. Video streaming with long latency, stall events, or bad image quality could impair the doctor's judgment, resulting in severe consequences. For example, when a stall event occurs in video streaming, the doctor may observe the screen freezing on the same image for a few seconds. This can make it difficult for the doctor to make an accurate diagnosis.

However, the quality of real-time video streaming in a moving ambulance may be unstable due to the fluctuating available bandwidth of the cellular network. There are two main factors. Firstly, the signal strength impacts the available bandwidth of the cellular

doi:10.6342/NTU202301100

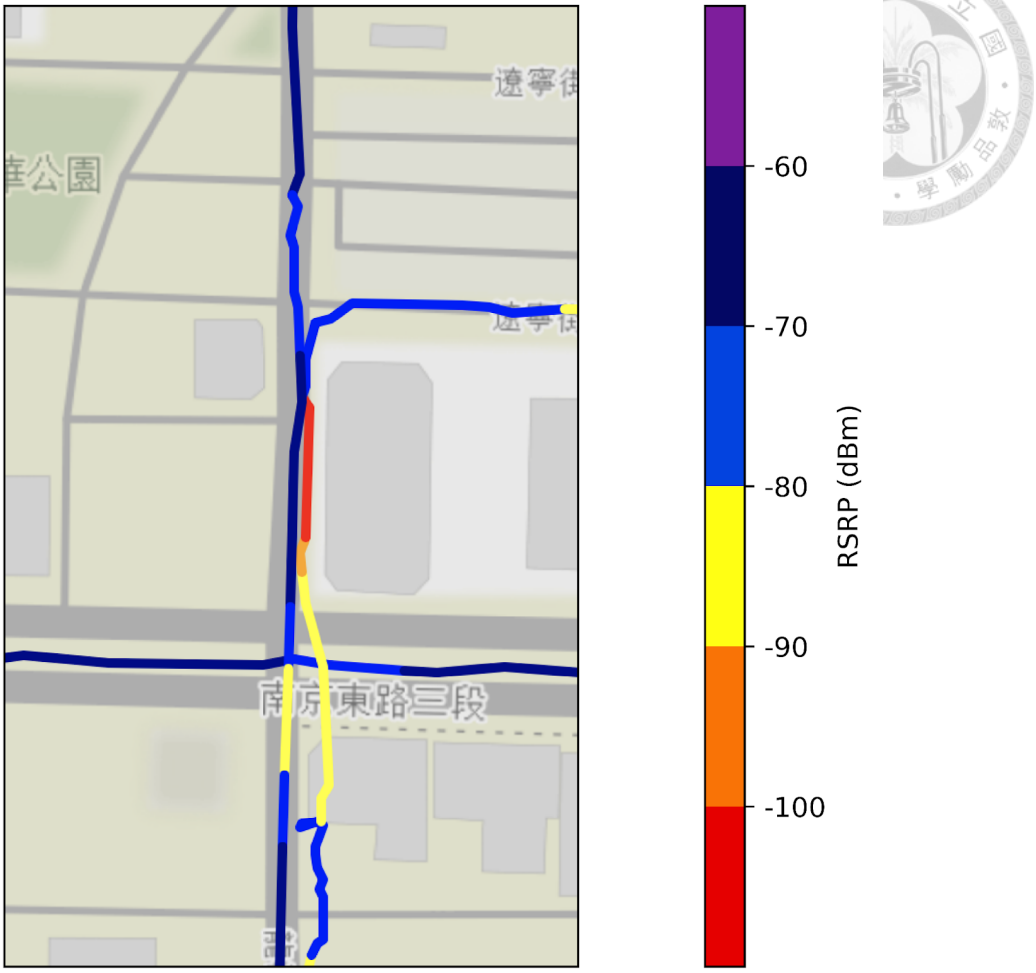


Figure 1.2: Signal strength change dramatically even just one lane apart.

network. The stronger the signal strength, the larger the available bandwidth of the cellular network, and vice versa. As shown in Fig 1.2, the RSRP value correlates with signal strength, with larger values indicating stronger signals. Although there is only one lane apart, the signal change significantly. Secondly, the base station's loading is related to the available bandwidth of the cellular network. Lee et al. [2] found that within the same base station, the number of devices is inversely proportional to the number of resource blocks allocated to each device. The more resource blocks allocated, the larger the available bandwidth, and vice versa. When the ambulance is moving, the phone may connect to base stations that are serving numerous users, leading to significant fluctuations in the available bandwidth of the cellular network. A sudden reduction of available bandwidth can produce

doi:10.6342/NTU202301100

Figure 1.3: Broken image.

blurry images, skipped frames, or video stalls, hindering the quality of experience (QoE). In this paper, we call all distorted images broken images, as shown in Fig 1.3. Therefore, it is crucial to prevent such occurrences when video streaming in a mobile ambulance.

In order to mitigate these effects, it is crucial to control the video encoding bitrate of our video streaming system. We should allow the video encoding bitrate to adapt to the variable bandwidth of the cellular network. GCC (Google Congestion Control) [3] is an adaptive bitrate algorithm which is proposed by Google that can make video streaming adaptive and real-time. However, the available bandwidth of cellular networks changes rapidly. GCC is not capable of adapting to the fluctuating available bandwidth of cellular networks, which results in a degradation of the receiver's QoE. This is because GCC is not specifically designed for cellular networks.

To design a video streaming system that is specific for cellular network, in this work, we propose a smart ambulance video streaming system, EMS real-time-communication (EMS-RTC), specifically designed for the needs of EMS. We leveraged key insights from

doi:10.6342/NTU202301100

the user study, identifying factors that can impair the doctor's diagnosis or assessment. Different from video streaming services that are usually for entertainment purposes, to accurately diagnose a patient in EMS from streamed video, lower packet delay and jitter are of utmost importance. Accordingly, we leverage deep learning techniques to develop a real-time ABR model. The model is trained to adjust the bitrate to minimize these factors, such that the videos delivered to the doctor at the hospital are low-latency and without stalls.

We realize EMS-RTC with three main elements. First, we build a video streaming platform on the WebRTC framework [4, 5], one of the de facto methods for real-time streaming applications. Our video streaming platform can record both the transmitted and the received videos and per-packet information, which can be used for evaluation purposes or further processed to serve as the training and testing data of our model. Second, we compose a real-world data set, collected as we stream videos while driving the test car with the streaming platform across different areas in the city. We derive most of our features based on the collected per-packet information obtained from frequently-used feedback, i.e., transport-wide RTCP feedback message [6], sent by the WebRTC receiver. In addition to these features, we also include a value quantifying the received power in the cellular network. Note that no additional equipment is needed for our system to perform bitrate adaption, thus reducing the system's complexity. Finally, we develop a Long Short Term Memory (LSTM) network to infer a proper bitrate adjustment decision. To avoid sudden large changes in bitrate that break video smoothness, the LSTM network serves as a classifier to determine whether the bitrate should be increased, decreased, or kept, instead of directly determining the optimal bitrate.

For comparison, in this paper, we also implemented a state-of-the-art ABR solution,

doi:10.6342/NTU202301100

WebRTC with Google Congestion Control (GCC). Both WebRTC with GCC and EMS-RTC are evaluated in real-world driving scenarios. The key contributions of this work include:

- We referred to the user study conducted by Wang *et al.* [7] which identified the important factors influencing doctors' diagnosis of medical conditions. Based on this, we designed EMS-RTC, a bitrate adaptation model.
- EMS-RTC is able to reduce the total time duration of stall events by a factor of 3.85 with respect to GCC.
- EMS-RTC is able to achieve similar video quality in terms of structural similarity (SSIM) with an average bitrate of 4600 Kbps, compared to 6100 Kbps used by WebRTC with GCC.

doi:10.6342/NTU202301100

Chapter 2 Related Work

2.1 Related Work

In this section, we briefly discussed previous works on ABR algorithms. Some ABR algorithms are rule-based. Kurdoglu *et al.* [8] presents a ruled-based bitrate control algorithm based on predicted bandwidth with a linear adaptive filter. Kreith *et al.* [9] presents a Forward Error Correction (FEC)-based bitrate adaptation method that defines network states and utilizes these states to determine how to control the bitrate and FEC rate. Carlucci *et al.* [3] utilizes packet loss rate and one way delay variation to adjust video encoding bitrate with Kalman filter. While these algorithms have demonstrated fine performance, such algorithms are tailored for video-watching experience but not optimized for EMS on the ambulances.

There also exist some reinforcement learning-based ABR algorithms. Mao *et al.* [10] trains the reinforcement learning agent in a simulated environment and gets a great performance. Zhou *et al.* [11] adjusts the bitrate by an imitation learning model according to the feedback report from a WebRTC receiver. However, the performance is only evaluated in a low bitrate range ($<$1.2 Mbps). Zhang *et al.* [12] develops an online reinforcement learning-based adaptation framework. It uses a hybrid learning mechanism, fusing reinforcement learning and GCC in WebRTC to improve its robustness. Nevertheless, re-

doi:10.6342/NTU202301100

inforcement learning is usually a computation-intensive model, which will significantly increase the construction cost if applied in the smart ambulance. Our work aims to bridge the gap between simulation and reality and develop a cost-efficient system that applies to smart ambulance applications. In addition, Yan *et al.* [13] found two things. First, to make training reinforcement learning agents easier, a trace-driven simulator allows experimenters to eliminate statistical noise, such as poor network conditions. However, this approach may lead to suboptimal performance of the agents in the real world. Second, it is challenging for the same agent to adapt to different network conditions across various locations. These factors make it challenging to train reinforcement learning agents to adjust bitrate.

Some works combine machine learning and cellular information. Yue *et al.* [14] utilizes random forest machine learning algorithms to predict the downlink throughput of LTE networks based on LTE layer information. The model is validated using data from various service providers and mobile devices. The study leverages QxDM [15] to collect LTE layer information and compares the granularity differences between QxDM and the Android API. However, their primary focus is on predicting downlink throughput rather than uplink throughput. Lee *et al.* [2] presents an ABR method based on uplink throughput prediction. Fine-grained lower layer information in the LTE network obtained by an additional LTE monitoring tool is used as the input to a 2-stage LSTM network. Although their work effectively prevents video stalling, it requires installing MobileInsight [16, 17]. Installing MobileInsight requires rooting the phone, which may cause additional overhead. Moreover, MobileInsight only supports specific Android phones. It is not available on all commodity phones. To mitigate overhead of installing MobileInsight, we utilize Android API to obtain signal information. Although the Android API cannot provide signal infor-

doi:10.6342/NTU202301100

mation with the same level of granularity as MobileInsight, it can significantly reduce the overhead of installing MobileInsight.

doi:10.6342/NTU202301100

Chapter 3 Background

3.1 H.264 frame

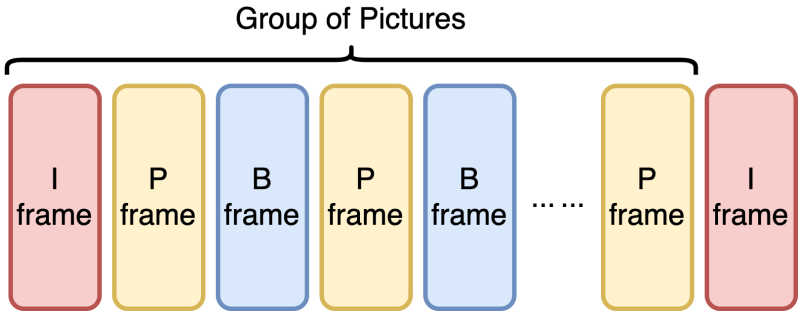


Figure 3.1: Group of Pictures

We choose H.264 [18] as our video coding scheme, because it has less encoding time and better compression ratio than VP9 [19]. The video encoding bitrate can be regarded as the compression rate. When video streaming, using a higher bitrate to encode a video results in better video quality. However, it also leads to higher bandwidth consumption. On the other hand, encoding a video with a lower bitrate results in a lower video quality, but it also reduces the bandwidth usage. In our work, it is necessary to control the bitrate in order to adapt to the bandwidth of the cellular network. This requires finding a compromise between video quality and bandwidth usage.

In H.264 video encoding, there are three types of frames: I-frames, P-frames, and B-frames. The I-frame is considered the most important frame, as P-frames and B-frames

doi:10.6342/NTU202301100

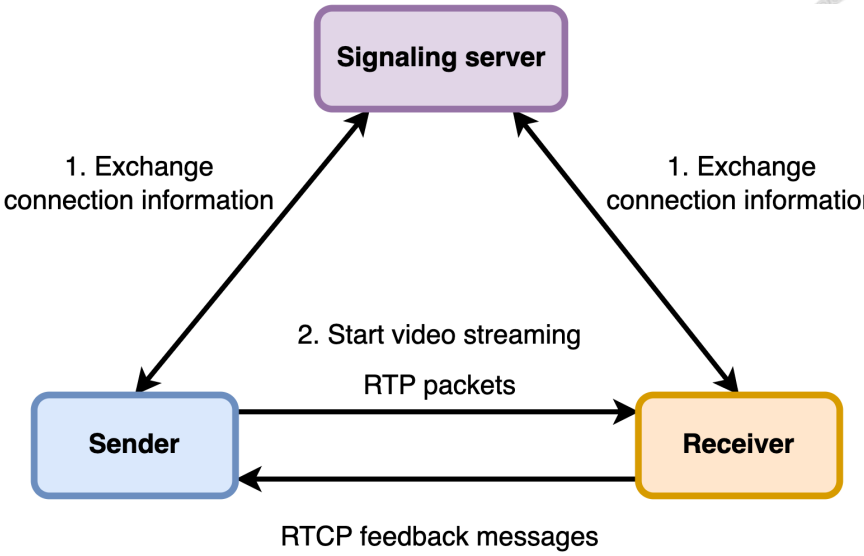


Figure 3.2: Measured RSRP CDF when streaming video using GCC and EMS-RTC.

require an I-frame for decoding. B-frames, specifically, rely on both preceding and subsequent frames for decoding. As shown in Fig 3.1, we refer to the P-frames and B-frames between the I-frame and the next I-frame as the Group of Picture (GOP). In the event of packet loss during video streaming, it can lead to broken images that persist until the end of the GOP. This is because the other frames in the GOP depend on the broken frame, leading to error propagation. Thus, the length of the GOP can impact the duration of the broken images. To mitigate the time of broken images in video streaming when packet loss, we set the GOP length in our video streaming to 30.

3.2 Web Real-Time Communication(WebRTC)

Web Real-Time Communication (WebRTC) [4] is a commonly used Web APIs for building audio and video streaming applications, such as voice/video conferences and live video/voice chats. WebRTC is based on Real-time Transport Protocol (RTP) and Real-time Transport Control Protocol (RTCP) protocol. RTP is a protocol for video/audio streaming, and RTCP is a protocol to transmit streaming status information, for example,

doi:10.6342/NTU202301100

jitter, one-way delay, packet loss rate, etc., to let the sender know the network status.

In general, a WebRTC system consists of a sender, a receiver, and a signaling server. As the Fig 3.2 shows, when the sender and receiver attempt to establish a connection, they encapsulate the necessary information for the connection within the Session Description Protocol (SDP), which is used to describe multimedia communication sessions. They send SDP to the signaling server, which facilitates the exchange of SDP between the sender and receiver to establish the connection. Once the connection is established, the sender starts to transmit RTP packets to the receiver. The RTP packets contain multimedia content. Besides, every RTP packet has its sequence number. The receiver statistics the multimedia streaming status puts the statistic into RTCP packets and sends it back to the sender to let it know the streaming status to adjust the encoding bitrate.

3.3 Google congestion control (GCC)

Google congestion control (GCC) [3] is a popular bitrate control algorithm. The mechanism consists of a delay-based and a loss-based controller, usually implemented in the sender. GCC determines the target bitrate based on the packet loss rate when the smoothed one-way delay variation exceeds an adaptive threshold. The receiver periodically sends per-packet feedback to the sender at a configurable interval, using a Transport Wide RTCP feedback message called Transport Wide Congestion Control (TWCC) feedback [6]. The feedback contains the packet sequence number, whether the packet is received, and the packet arrival time. When the feedback message doesn't report any packets, it means that the receiver didn't receive any packets from the time the last feedback was generated to the time the current feedback is generated. Past research has indicated that the

doi:10.6342/NTU202301100

adaptive threshold mechanism in GCC results in longer time to react to frequent changes in bandwidth [20]. Moreover, using a large bitrate as the network condition quickly degrades is the main reason the video stream stalls. As bandwidth fluctuation is common in mobile scenarios, to allow better diagnosis performance in remote emergency care, quick adjustment of transmission bitrate in response to the changes of the network condition is crucial. Our system aims to achieve this by leveraging the information from the TWCC feedback messages to quickly determine the best transmission bitrate, replacing the GCC mechanism in a WebRTC system.

doi:10.6342/NTU202301100

Chapter 4 System Design

4.1 System Overview

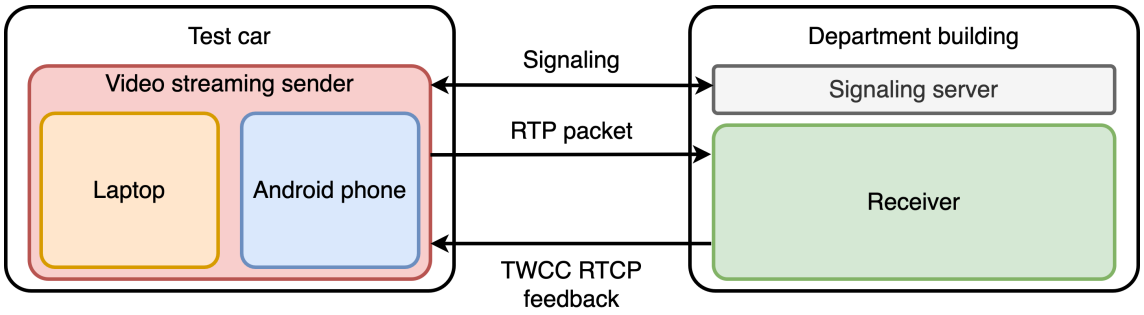


Figure 4.1: Overview of the video streaming platform.

To perform real-world measurements and experiments, we constructed a video streaming platform. The platform is to emulate the actual use case, where the sender in the ambulance continuously streams the video to the receiver in the remote monitoring center. Fig. 4.1 shows the implementation of our developed platform. To simulate an ambulance, we collected data in Taipei while driving a car. The sender in the car transmits video streaming over a 4G LTE network provided by an Android phone. The receiver and signaling server are located in our department building and connected by wired Ethernet connections. During the connection setup, the sender and receiver exchange their information through the signaling server.

Once the sender and receiver have established a connection, the sender begins to

doi:10.6342/NTU202301100

transport video frames captured by the webcam. We use H.264 codec [18] to encode video frames. On the sender side, as a frame is usually split into multiple packets, three additional bytes are added to every RTP packet. This is to convey the essential information for decoding the frames, such as the frame number that the packet belongs to. Upon sending the packets, the sender saves the packet departure time and packet size. The sender also stores all the received TWCC feedback messages sent by the receiver every 100 ms, including their arrival time and the packet arrival timestamps reported in the feedback message. The feedback contains the packet arrival timestamps. With these parameters, i.e., packet departure time, packet size, packet arrival time, and feedback arrival time, the network conditions at the moment the packet was transmitted can be fully assessed. We process feedback messages as features on a per-message basis. The sender can adjust to the optimal transmission rate using these as feature inputs (see Sec. 4.2).

On the receiver side, all received packets and their arrival times are saved. A video player is implemented to reconstruct the video frames from the received RTP packets. We can repeatedly utilize these packets after we save them. The received packets are placed in a jitter buffer according to their arrival times. The player starts the playback with a buffering time of 300 ms[1].This buffering time starts immediately when the packet containing H.264 SPS and PPS information, which is required for video decoding, arrives at the jitter buffer. Then, the player retrieves the packets from the jitter buffer and displays a frame every 1/30 seconds, as the frame rate of the webcam is 30 fps. For instance, if the k-th frame is displayed at time $T(k)$, the next frame $k+1$ should be displayed at time $T(k+1) = T(k) + \frac{1}{30}$ (s). For data collection and evaluation purposes, we save all decoded frames from the player for further processing.

[1]This is configured according to the recommendation of the International Telecommunication Union (ITU) that one-way delay should be below 400 ms in interactive applications [21].

doi:10.6342/NTU202301100

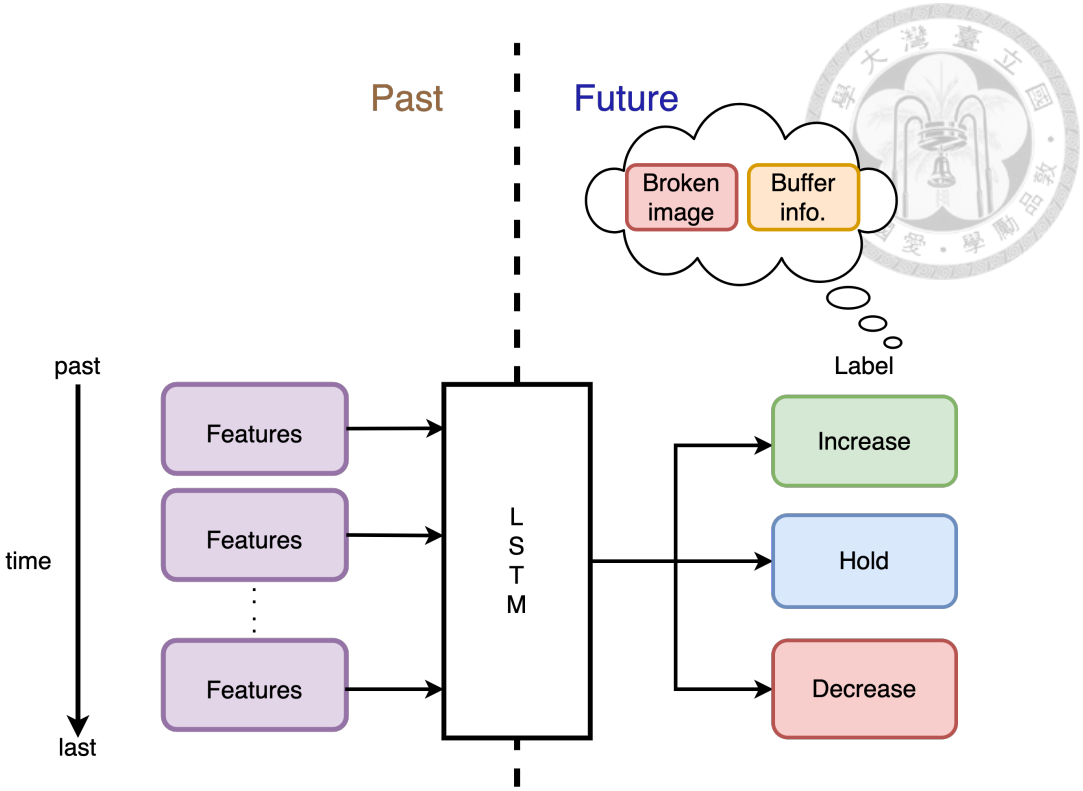


Figure 4.2: Bitrate adaptation model overview.

A *stall* event happens when no packet belonging to the current frame exists in the jitter buffer, i.e., at time $T(k+1)$, insufficient packets to decode frame number $k+1$ exist in the jitter buffer. Time $T(k+1)$ is then regarded as the start time of this stall event. During this stall event, the receiver continuously attempts to decode a new frame every 1/30 seconds. Once the player receives sufficient packets to put together a playable frame, the stall event ends. For example, if the receiver is able to collect sufficient packets for frame $k+3$ from the jitter buffer at time $T(k+3)$, the stall event ends. Then, the stall time is calculated as $T(k+3) - T(k+1)$.

Once we have processed the features and reconstructed the video streaming, the next step is to train our bitrate adaptation model. Fig. 4.2 shows our bitrate adaptation model overview. To make decisions about adjusting the bitrate, the bitrate adaptation model utilizes past network conditions and bitrate usage, which are included in the features. The model uses features obtained at different timestamps to predict how to adjust the bitrate.

doi:10.6342/NTU202301100

The model is designed to have three possible outputs: *increase*, *hold*, and *decrease* of the bitrate. *Increase* corresponds to a 5% increase of the bitrate, while *decrease* corresponds to a 10% decrease of the bitrate. The rates of increase and decrease are determined empirically. *Hold* would maintain the current bitrate without change. To determine the adjustment label, we utilize future video streaming quality. Future video streaming quality can infer from the severity of broken images and the number of frames remaining in the jitter buffer. These also represent our video streaming QoE. Finally, we take the future video streaming QoE to label our features.

We chose to use LSTM [22] as our bitrate adaptation model because the features we use are time series data. These features are derived from feedback messages, which arrive at the sender in a temporal order, much like a time series. To address this time series problem, we opted to use the LSTM model. In the following sections, we will introduce the selection of various features, the labeling approach, and the data collection process one by one.

4.2 Features

At the core of our proposed EMS-RTC video streaming platform is the bitrate adaptation model, which takes features outlined in this section in real-time as input to determine whether it is necessary to adjust the current bitrate. With such designs, the model is periodically executed such that the bitrate is adjusted to closely match the changing network condition. In the following, we provide a detailed description of how the model is developed.

doi:10.6342/NTU202301100

4.2.1 Model Features

In our video streaming system, the sender maintains a list of sent packet parameters: sequence number i, the departure time $S(i)$, and the packet size $P(i)$. When the sender receives a periodic RTCP report, the sender can determine whether the packet of sequence number i is received by the time the feedback was sent and, if so, when it arrives at the receiver $A(i)$. Using these parameters, we can derive the following features for model training:

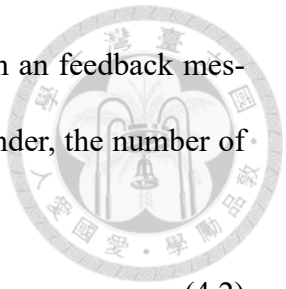
- **Byte in flight (BIF)** is the number of sent bytes that are not yet acknowledged by the receiver (i.e., reported in the feedback message). A large BIF value is likely a sign of network congestion since more packets are sent but stuck within the connection during the transmission. We obtain BIF from an RTCP report. When the sender receives a feedback message with sequence number m, a BIF value $B(m)$ is obtained by summing up the packet size of all the unreceived packets, given by

$$B(m) = \sum_{k=i_r+1}^{i_t} P(k), \tag{4.1}$$

  where i_r is the maximum sequence number among the received packets reported by the feedback message and i_t is the maximum sequence number of the transmitted packets.

- **Received throughput** is the amount of received bytes by the receiver in the past 100 ms, and can also be used as an indicator to know whether the packets are congested. When the received throughput is small, it is likely that the available throughput reduces due to bad network conditions. To calculate the received throughput,

doi:10.6342/NTU202301100



we obtain the set of received packet sequence numbers Q from an feedback message. Since the size of the received packets is known to the sender, the number of received bytes is given by

$$R(m) = \sum_{k \in Q} P(k), \tag{4.2}$$

where m is the sequence number of the RTCP report.

- **Packet loss rate** is a common feature to reflect the network conditions. It also relates to whether video streaming has broken images. From the feedback message, we can obtain the sequence numbers of the first and the last packet in the report to determine the total number of packets the receiver is supposed to receive N. The feedback can also be used to calculate the number of lost packets in this range N_L. The packet loss rate is then given by N_L/N.

- **One-way delay variation (OWDV)** has been used as a key indicator in GCC to perform bitrate control. OWDV is defined as the difference between two consecutive one-way delays. The reason for using OWDV over the one-way delay is its better accuracy since the sender and receiver are typically not synchronized. When the OWDV value gradually increases, it indicates that the available bandwidth of the cellular network may be decreasing, and vice versa. OWDV can be calculated from the difference between the inter-departure time and inter-arrival time of two consecutive packets, given by

$$\text{OWDV}(i+1) = [A(i+1) - A(i)] - [S(i+1) - S(i)], \tag{4.3}$$

where $A(i)$ and $A(i+1)$ are the arrival times of packet i and $i+1$, respectively, and $S(i)$ and $S(i+1)$ are the sent times of packet i and $i+1$, respectively.

doi:10.6342/NTU202301100

- **Feedback effectiveness** is defined as the effectiveness of a feedback message based on the time difference between when a feedback message is received and the time that the sender is ready to adjust the bitrate with that feedback message. Feedback messages received more recently with respect to the bitrate change time should be more representative of the ongoing network conditions.

- **Reference Signal Received Power (RSRP)** indicates the received strength of the signal of the connected base station captured by the mobile device, reflecting the LTE channel quality. It is one of the key factors determining the Modulation Coding Scheme (MCS) and thus the available throughput. When the signal is weak, the transmission will use simpler modulation, leading to a decrease in throughput, and vice versa. To observe RSRP variation with mobility, the value measured by the cellphone is captured every 500 ms.

- **Bitrate** is defined as the bitrate usage when the sender receives a feedback message. The video encoding bitrate is an indicator of the amount of data transmitted in the past. This allows our model to determine whether the current bitrate is appropriate for the network condition. Transmitting more data than what can be supported by the link, especially when the link is close to saturation, could induce or exacerbate network congestion.

doi:10.6342/NTU202301100

4.3 Post processing

4.3.1 Maintaining a consistent sampling rate.

In our system, most parameters (i.e., BIF, received throughput, packet loss rate, and feedback effectiveness) are derived according to the transport-wide RTCP feedback message generated every 100 ms. As a result, the sampling rates of OWDV and RSRP need to be adjusted to match the other features. For OWDV, which is a feature that changes per packet, we sum up the OWDV of the packets reported in the feedback message to have one OWDV value per feedback. For RSRP and bitrate, the last value acquired before the feedback message is used as the sampled RSRP and bitrate value when the feedback message is received.

4.3.2 Mitigating data fluctuation caused by the size of the sent frame.

During our experiments, we found that BIF and received throughput sometimes *increase sharply together*, which is counter-intuitive. In general, these two parameters should have a negative correlation, i.e., a bad network condition results in *large* BIF and *small* received throughput and vice versa. Nevertheless, we discovered this abnormal situation happens because of the transmitted frame size and bitrate variation. In H.264 encoding, video frames are encoded to I-frame, P-frame, and B-frame. The size of the I-frame is much larger than the P- and B- frames. During the I-frame transmission, the BIF and received throughput grow dramatically due to the induction of packets. As shown in Fig. 4.3, the orange line is the bitrate value, and the blue line is the BIF value. The top figure exhibits numerous small peaks, which are likely caused by the influence of I-

doi:10.6342/NTU202301100

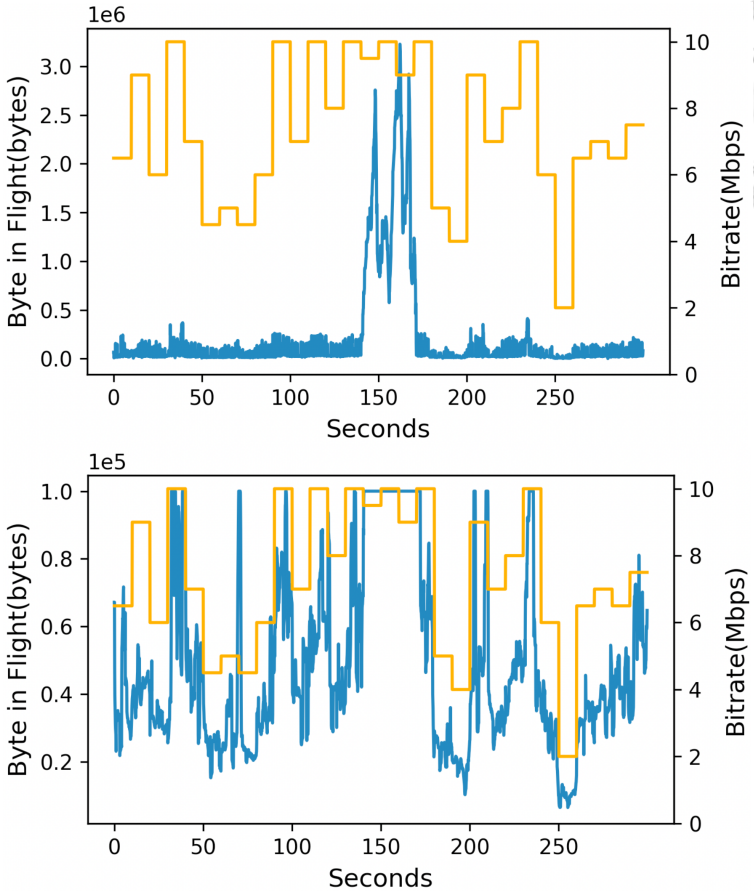


Figure 4.3: Bitrate and I frame influence on BIF and Received throughput value

frames. Likewise, when using a higher bitrate to encode the video frames, the BIF and received throughput values also increase sharply. As shown in Fig. 4.3, the bottom figure demonstrates how the bitrate affects BIF. Such value increment would confuse the model, hindering its performance. To mitigate the influence caused by the frame type and the bitrate, we first normalize the size of each packet by the bitrate. Then, BIF and received throughput are recalculated using Eq. 4.1 and Eq. 4.2, respectively. Finally, we calculate the moving average of BIF and received throughput within one second to smoothen the peak values caused by the I-frame. This is because, in our case, the transmission of the I-frame occurs once per second.

doi:10.6342/NTU202301100

4.4 Quality of service metric and data labeling

In [7], Wang *et al.* show that stalling and image quality are the two most important factors influencing remote diagnosis. Moreover, the former is much more important than the latter. Here, we aim to create a metric indicative of the declining quality of service in the near future, denoted by $\tilde{v}(\cdot)$. Such a metric can be used to label the data with proper output, i.e., how the bitrate should be adjusted to maintain a good video-watching experience. To this end, the following two metrics are selected:

- **Structural similarity (SSIM)** [23] , denoted by $M(\cdot)$, is a common index to measure the similarity between two images, quantifying how much distortion exists between the initial image and the image degraded by the communication. To evaluate the video degradation, including broken frames and glitches in a frame, we calculate the SSIM value of each frame using the original sent video and the corresponding received video.

- **Buffer occupancy**, denoted by $O(\cdot)$, is defined as the number of playable frames in the jitter buffer and given by subtracting the frame number that is currently in play from the maximum frame number of the packets in the jitter buffer. When the buffer occupancy reduces to zero or less, the jitter buffer no longer contains packets of any playable frame. Negative buffer occupancy values imply that the frame which should be played at the moment has a much larger frame number than that of the received packets. This indicates a stall event. Hence, buffer occupancy can be regarded as a leading indicator to stall events.

Then, to create the indicative metric $\tilde{v}(\cdot)$, we first normalize both to the range between

doi:10.6342/NTU202301100

0 and 1. As stalling is more critical than image quality, we empirically set the weights of SSIM and buffer occupancy to 0.25 and 0.75, respectively. In [7], Wang *et al.* found that stall events have a greater impact on diagnosis than broken images, as informed by paramedics and doctors. Therefore, buffer occupancy is given more importance. Then, using weighted sum, the value of the metric for frame j is given by

$$v(j) = 0.25 \cdot M(j) + 0.75 \cdot O(j), \tag{4.4}$$

where $M(j)$ is the SSIM of the j-th frame and $O(j)$ is the buffer occupancy when playing the j-th frame.

To label the data corresponding to model prediction time t_{pred}, i.e., the time to execute the model and determine whether rate adjustment is needed, we obtain the set of the frames played between t_{pred} and $t_{\text{pred}} + 1$ (s), denoted by F. Using Eq. 4.4, an average of $v(j)$ of these frames are calculated, given by

$$\tilde{v}(t_{\text{pred}}) = \frac{1}{|F|} \sum_{j \in F} v(j). \tag{4.5}$$

The distribution of $\tilde{v}(\cdot)$ formed by the collected empirical data has the median at 0.96. When $\tilde{v}(\cdot)$ is greater than or equal to the median, it is likely that the bitrate can be increased without affecting the QoE. In addition, we investigated the distribution of $\tilde{v}(\cdot)$ formed only by the data collected one second before a stall event, and the 75-percentile value is at 0.93 in this distribution. It is likely that when $\tilde{v}(\cdot)$ is less than this value, stalling is imminent and a decrease in the bitrate is crucial to either prevent stalling or expedite the recovery of streaming when the network condition improves.

Accordingly, we label the collected data with these criteria:

doi:10.6342/NTU202301100

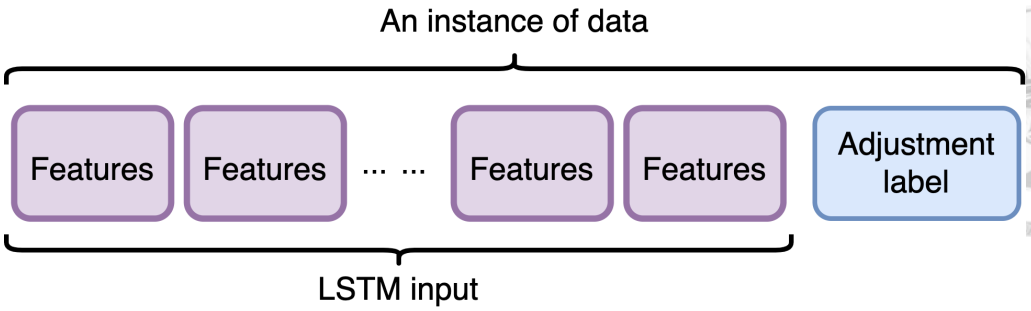


Figure 4.4: An instance of data.

- Increase: $\tilde{v}(t_{\text{pred}}) \geq 0.96$ and the bitrate is less than or equal 6 Mbps.
- Hold: $0.93 \leq \tilde{v}(t_{\text{pred}}) < 0.96$ or when $\tilde{v}(t_{\text{pred}}) \geq 0.96$, but the bitrate is higher than 6 Mbps.
- Decrease: $\tilde{v}(t_{\text{pred}}) < 0.93$

4.5 Model

The input of the proposed model comprises seven features: the seven features introduced in Sec. 4.2 and the current bitrate. EMS-RTC executes the model every 200 ms. At the bitrate prediction time t_{pred}, the seven features are obtained from the ten most recent feedback messages received between time $t_{\text{pred}} - 2$ (s) and t_{pred}. We assign the adjustment label which obtained in Sec. 4.4 to the features. An instance of data is shown in Fig. 4.4, with a data shape of 10×7. Finally, the model output can be used to determine whether the transmission bitrate should be adjusted. Normally, these ten feedbacks correspond to packets received in one second, as the feedback is sent every 100 ms. However, when the network condition is poor, the feedback message is often lost. If such a case happens and the sender has less than ten feedback messages at the time the model should be executed, the system would hold the current transmission bitrate. If the same consecutively

doi:10.6342/NTU202301100

happens, the transmission rate would be decreased in the same manner when the model outputs “decrease”.

4.5.1 Training data collection

To collect data for model training, we operated the video streaming platform as described in Sec. 4.1, but used pre-recorded and pre-encoded videos instead of the live capture from the webcam. This is to ensure that the receiving delay and quality are only affected by the mobile network quality but not by the encoding process before the transmission. To do so, we recorded three videos of emergency drills in a moving ambulance. Each video lasts about five minutes. The videos for video streaming are pre-encoded by the H.264 encoder. In H.264, the B-frame needs to rely on both preceding and subsequent frames for decoding. To ensure real-time decoding for every frame, our video does not encode any B-frames. The video is encoded with bitrates that change every ten seconds and in the range of 1 to 10 Mbps. We generated 20 videos with different transmission bitrate sequences for each video, producing 60 videos in total. During the experiments, we randomly select a pre-encoded video for streaming.

We performed the data collection in Taipei, Taiwan, covering some downtown and suburban areas as well as some mountain areas, as depicted in Fig. 4.5. To consider the potential influence of weather and time (e.g., rush hours and off-peak periods), the measurements were conducted on four different days and both sunny and rainy days. A total of approximately nine hours of data was collected, corresponding to over ten million RTP packets. After processing, we obtained a dataset consisting of 166,548 instances of data. Detailed information of data collection is in Table 4.1

doi:10.6342/NTU202301100

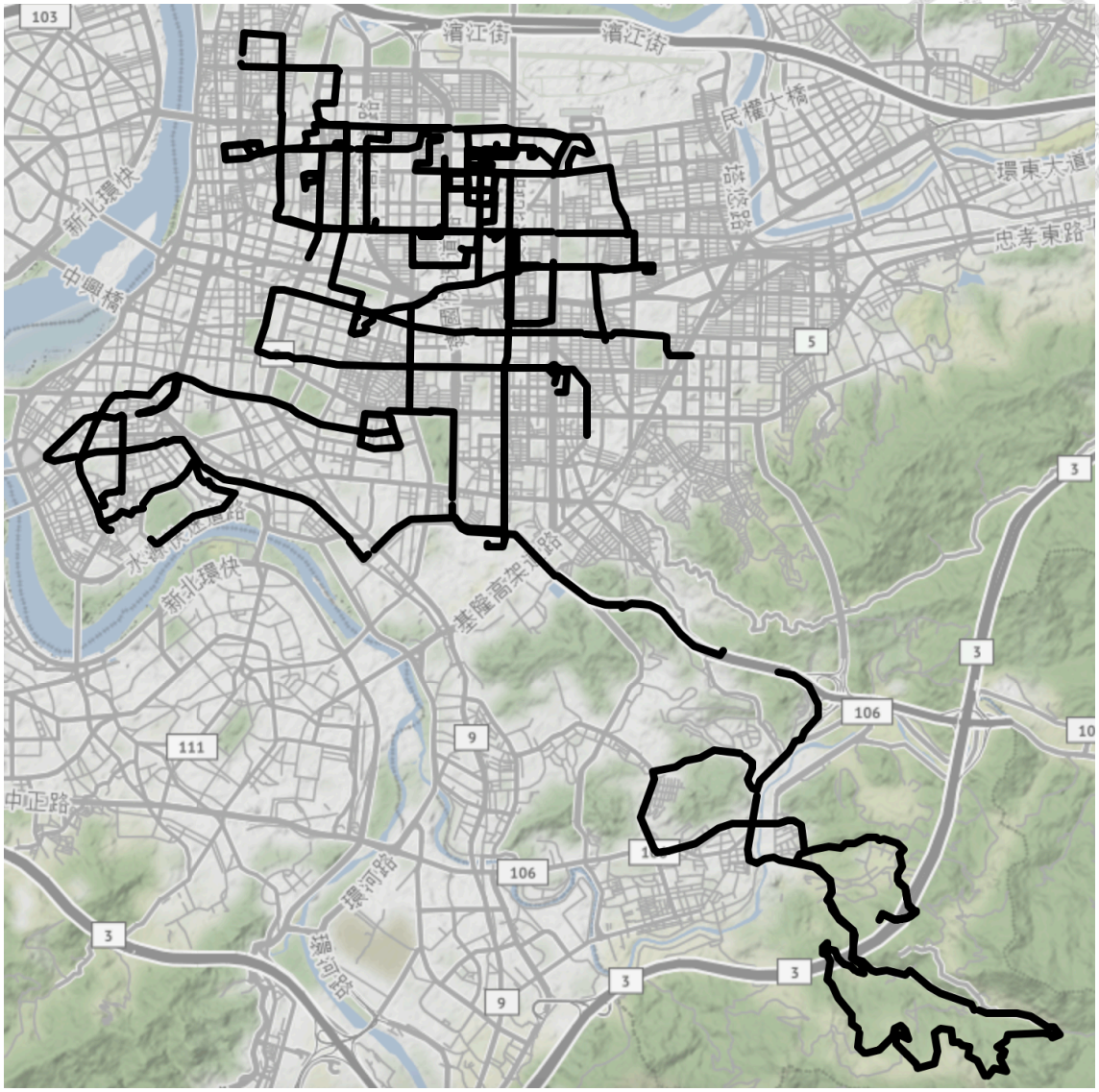


Figure 4.5: The data collection GPS traces.

4.5.2 Model architecture and training

We adopt an LSTM-based deep learning approach to extract the characteristics of network conditions and bitrate variation over time. Fig. 4.6 shows our model architecture. The first layer is a single LSTM layer with 32 hidden units. Then, a dropout layer is added before the fully connected layer to prevent over-fitting. The reason for using only 32 hidden units is to prevent over-fitting.

To ensure the balance within data with different labels, we randomly sample 15000

doi:10.6342/NTU202301100

Table 4.1: Details of data collection.

Total time	Speed	Time
9h 2m	0 to 90km/h	Afternoon, Night
Weather	Service provider	Location
Sunny, Rainy	Chunghwa Telecom	Downtown, Suburban, Mountain

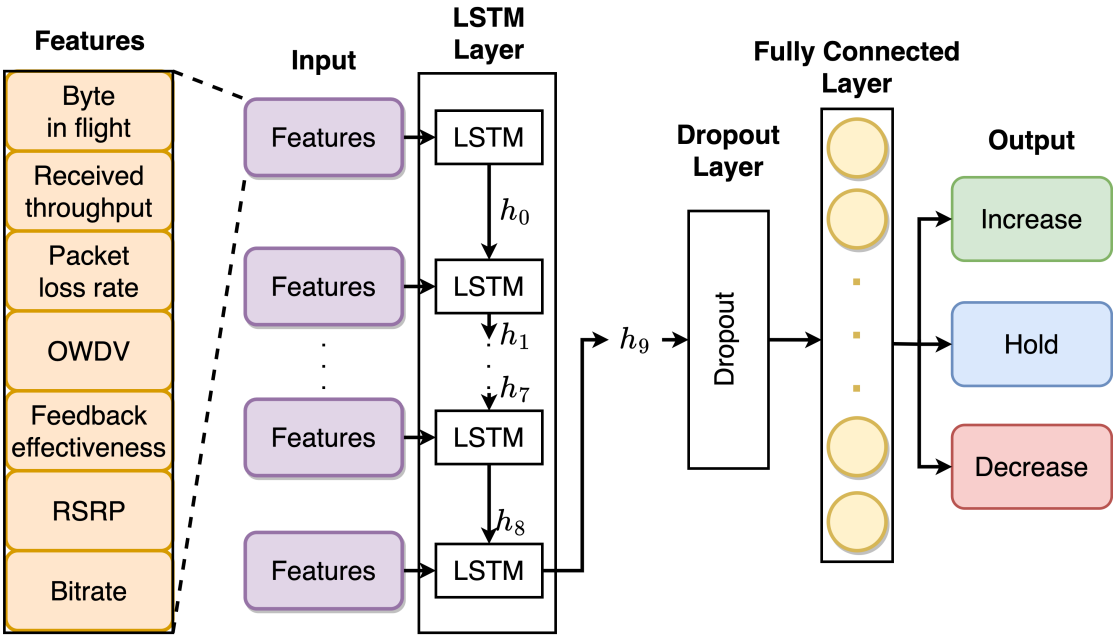


Figure 4.6: Model architecture.

instances of data from each class. Since our model is a classification model, we adopt cross-entropy as our loss function. We use Adam optimizer with an initial learning rate of 0.0001. Then, we divide the 45,000 instances of data into 5 folds for 5-fold cross-validation, with 80% used as the training set and 20% as the testing set. We select the fold that exhibits the best performance on the testing set as our final model. The accuracy and loss on the testing set are 89.78% and 0.2845, respectively. There are 2.93% data underestimated and 7.29% data are overestimated. The confusion matrix is in Fig. 4.7. Although we achieved excellent results on the testing set, the crucial factor is the performance of video streaming while a car is mobile in Taipei. We show the evaluation results in Ch 6.

doi:10.6342/NTU202301100

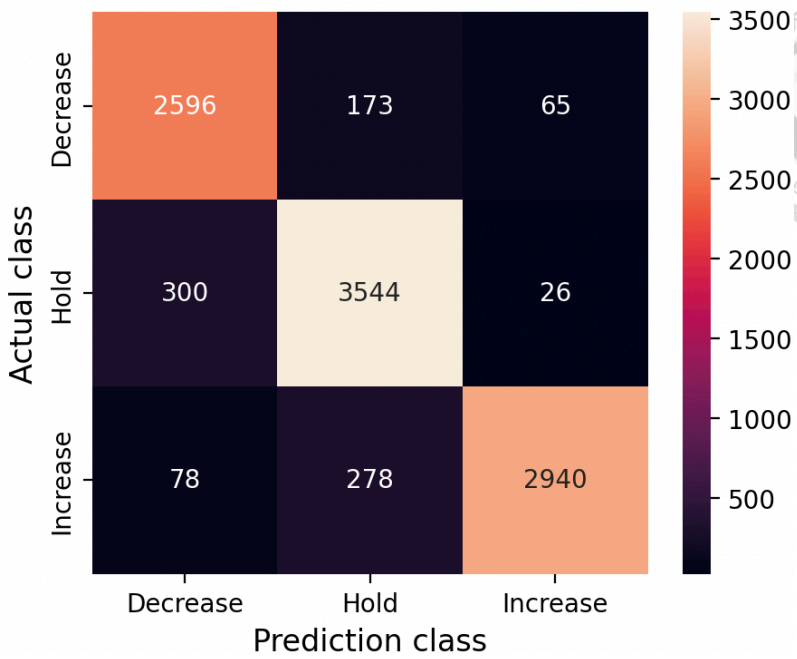


Figure 4.7: Confusion matrix of testing set.

Permuted feature	Accuracy	Loss
Byte in flight	42.96%	3.6583
Received throughput	86.74%	0.3896
Packet loss rate	88.1%	0.3581
One-way delay variation	65.96%	0.9713
Feedback effectiveness	81.18%	0.4717
RSRP	87.82%	0.3425
Bitrate	62.28%	1.644

Table 4.2: The impact of permuting one feature on testing accuracy and loss.

4.5.3 Features importance

To evaluate the impact of each feature on training, we employed the permutation importance method [24]. Using a trained model, we randomly permuted one feature in the testing set data to observe its impact on accuracy and loss, in order to determine its feature importance. Table 4.2 shows the permuted feature and its impact on testing accuracy and loss. We found that byte in flight, OWDV, and bitrate are very important. Surprisingly, RSRP appears to be less important, and we speculate that this may be due to insufficient granularity in RSRP measurements.

doi:10.6342/NTU202301100

Chapter 5 Implementation

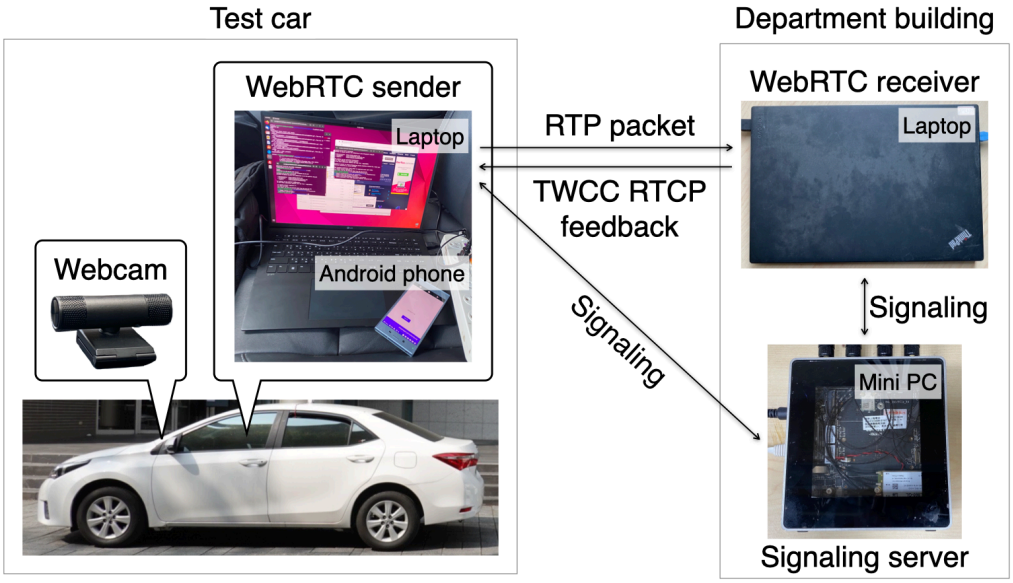


Figure 5.1: Overview of the video streaming platform.

In this chapter, we introduce our platform implementation for evaluation. We construct our video streaming platform using Pion WebRTC API. We have a sender, a receiver, and a signaling server, as Fig. 5.1.

5.1 Sender

The sender is an LG gram laptop with an Intel i5 1240P processor. The webcam is plugged into the laptop. The webcam is Avermedia PW315 1080p resolution webcam. The sender uses the LTE network to send data. The Sony Xperia XZs Android phone

doi:10.6342/NTU202301100

Figure 5.2: Battery.

shares the LTE network with the sender laptop by USB tethering. Besides, it transports RSRP information to the sender with Android Debug Bridge (ADB). The LTE network provider is Chunghwa Telecom[1]. There has a battery that supplies power for everything in the Altis, as Fig. 5.2. When evaluating, we drove the Toyota Altis car in Taipei. Our driving route included main roads as well as narrow alleys.

5.2 Receiver and Signaling server

The receiver, located in our department building, is a Thinkpad laptop equipped with an Intel i5 7200U processor. It connects to the internet using a wired connection. Similarly, the signaling server, also present in our department building, is a Mini PC powered by an

[1] At the time of writing, Chunghwa telecom is the biggest telecommunication service provider in Taiwan.

doi:10.6342/NTU202301100

Intel Celeron processor and also connected to the internet via a wired connection. The receiver is responsible for receiving the video streaming transmitted by the sender and storing all the packets of the video streaming.

When the EMS-RTC sender initiates video streaming, the initial encoding bitrate is set at 2000Kbps. Subsequently, the sender adjusts the encoding bitrate every 200ms.

doi:10.6342/NTU202301100

Chapter 6 Evaluation

6.1 Methodology

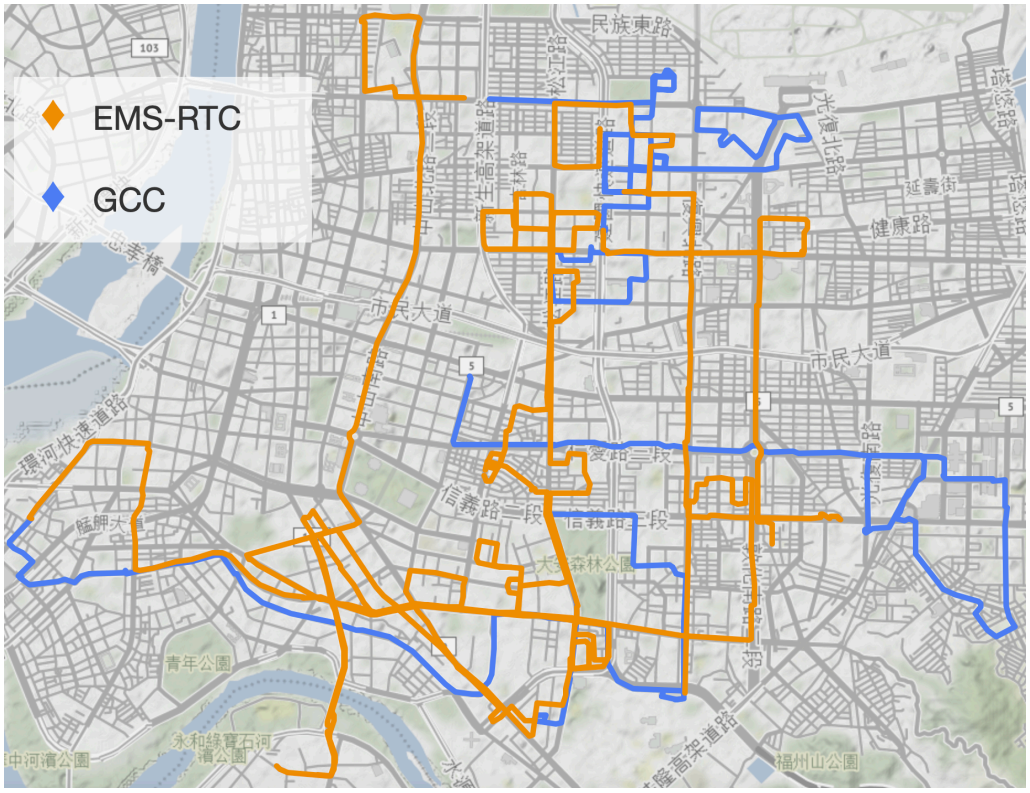


Figure 6.1: Driving route in Taipei, Taiwan.

The experiments were carried out with the same procedures outlined in Sec. 4.1, where video captured by the webcam is streamed to the receiver in real time over the 4G LTE network. However, the bitrate adaption model is periodically executed to determine whether the bitrate should be adjusted according to the network condition. The bitrate always falls between 1 and 7 Mbps. For comparison with the proposed EMS-RTC, we

doi:10.6342/NTU202301100

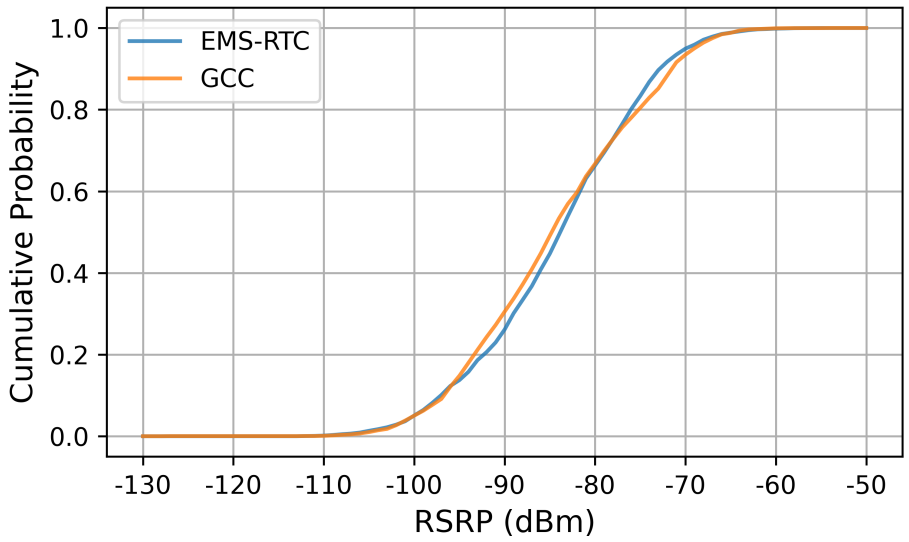


Figure 6.2: Measured RSRP CDF when streaming video using GCC and EMS-RTC.

Stall time (s)	0~0.5	0.5~1	1~1.5	1.5~2	2~2.5	2.5~3	3~3.5	3.5~4	4~4.5	4.5~5	≥5
GCC	166	18	19	9	10	6	5	3	4	5	24
EMS-RTC	36	8	6	5	3	2	2	1	0	2	5

Table 6.1: Number of stall events

implemented the GCC algorithm in our platform as a baseline. Although GCC doesn't utilize signal information, in [2], GCC performs even better than some machine learning algorithms that use signal information as features. Similar to the training data collection process, we drove the test car equipped with the sender in Taipei. Fig. 6.1. shows the routes driven during the experiments, where the experiments for GCC and our proposed EMS-RTC ran for 3.22 and 4.12 hours, respectively.

To ensure a fair comparison, we look into the cumulative distribution function of the received RSRP of the routes, as shown in Fig. 6.2. The durations that RSRP drops below -85 dBm when using GCC and EMS-RTC are 5126.5 s and 5241.4 s, respectively. One can see that despite the different routes taken when experimenting with GCC and EMS-RTC, their distributions are quite similar.

doi:10.6342/NTU202301100

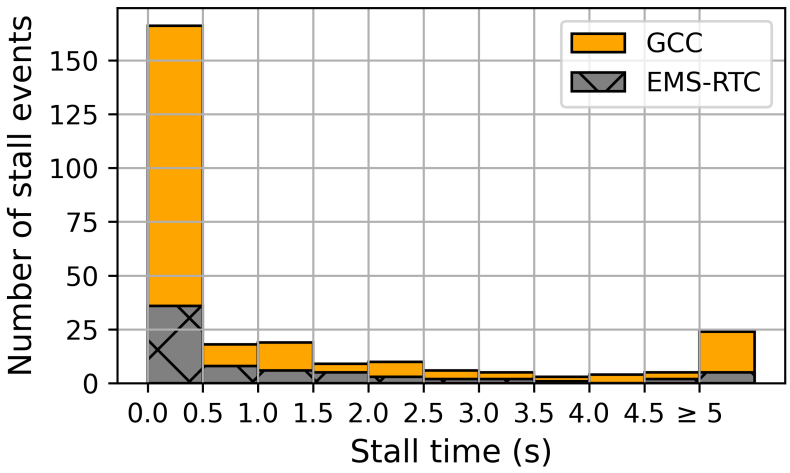


Figure 6.3: Comparison of the number of stall events.

6.2 Comparison of GCC and EMS-RTC

Comparison of stall time. During the real-world experiments, the total numbers of stall events using GCC and EMS-RTC are 269 and 70, respectively, corresponding to total stall times of 410.6 s and 106.6 s. Fig 6.3 compares the histograms of the number of stall events and the stall duration when using GCC and EMS-RTC. Table 6.1 shows the duration of stalls and the number of stall events. One can clearly see that EMS-RTC outperforms GCC, with fewer stall events for all values of stall time. It shows that our model can successfully forecast the bitrate adjustment and decrease the bitrate before the network environment deterioration. In Fig 6.4, we can observe the adjustment of the video encoding bitrate by GCC. It keeps a high bitrate even when the signal strength is weak, which can result in longer stall events if the network's available bandwidth is insufficient. According to Guo *et al.* [25], packets cannot be sent to the base station when the signal is weak. Instead, they are stored in the LTE firmware buffer. The more packets stored in the firmware buffer, the longer it takes to consume the packets and recover to a playable state from the buffer. As shown in Fig 6.4, the receiver receives the data transmitted at the black arrow when the time is at the red dot. The time gap indicates one-way delay

doi:10.6342/NTU202301100

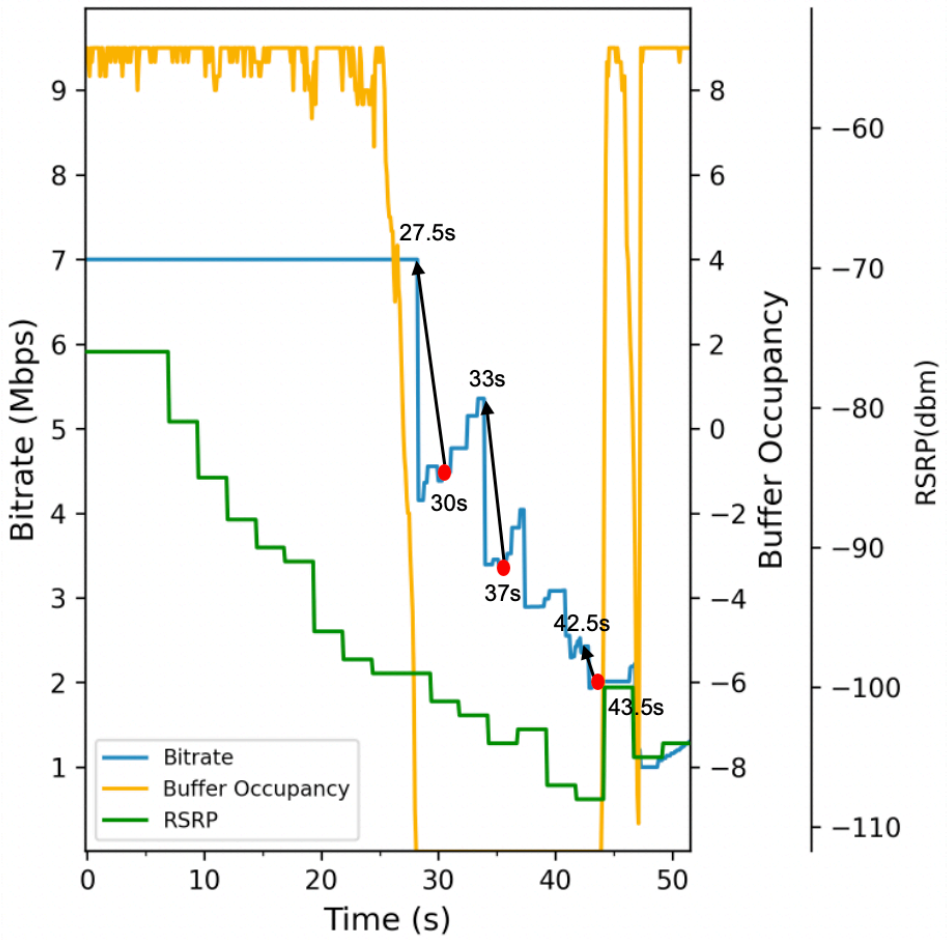


Figure 6.4: GCC maintains a high bitrate when the signal strength is weak may result in a longer stall event.

between the sender and the receiver. Despite GCC gradually decreasing the bitrate, but the one-way delay is still higher than 2 seconds. If a higher bitrate is maintained during a rapid deterioration of network conditions, the video streaming may stall. The stall time may be determined by the bitrate usage before the stall event.

One factor that we believe contributes to the significance of GCC's stall events is their lack of utilization of LTE information for bitrate prediction. This limitation prevents GCC from adapting to the rapid variance in cellular network available bandwidth, ultimately resulting in QoE degradation. As shown in Fig 6.5, EMS-RTC adjusts the encoding bitrate adaptive to RSRP. It mitigates the stall event occurring. The occurrence of the stall event in Fig 6.5 at around 24 seconds is attributed to a weak signal. Despite EMS-RTC adjusting

doi:10.6342/NTU202301100

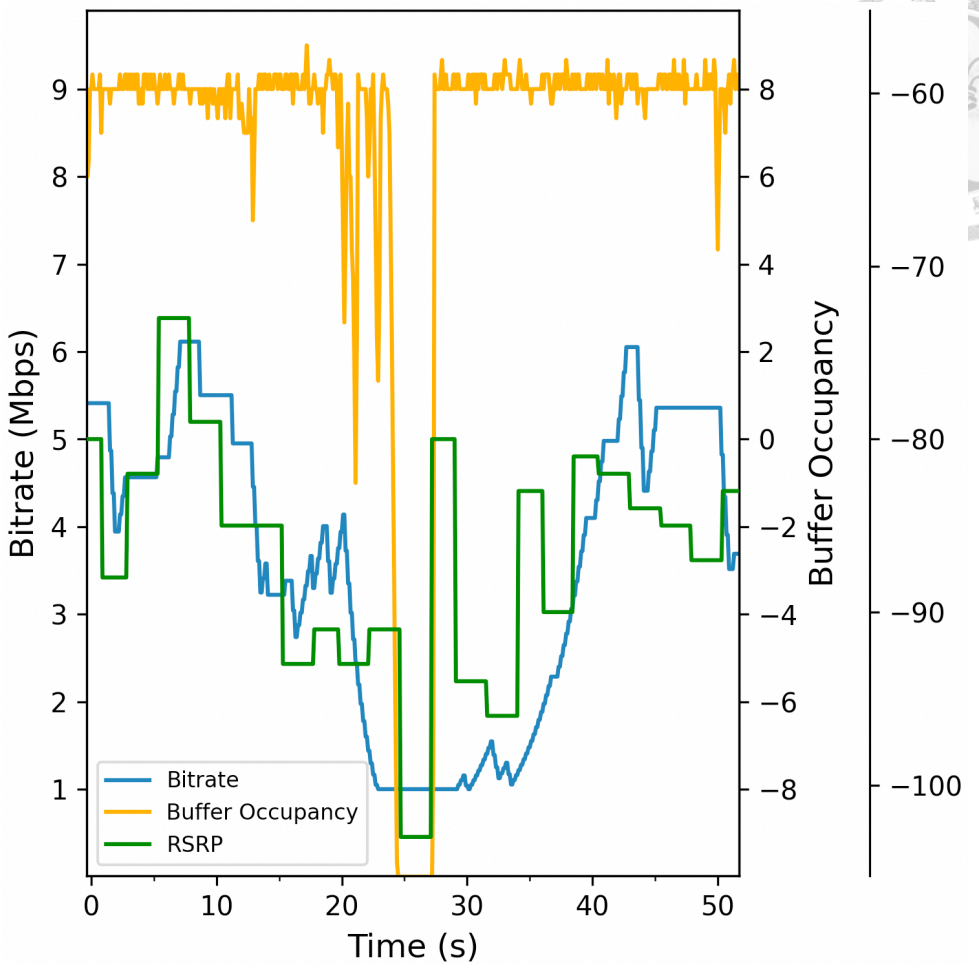


Figure 6.5: EMS-RTC adjusts bitrate adaptive to signal strength.

to the lowest bitrate of 1000Kbps, the stall event still happens. This can be attributed to insufficient base station coverage, leading to the occurrence of the stall event. While the signal strength improves, EMS-RTC gradually increases its bitrate.

Comparison of one way delay. In Fig. 6.6, the average one-way delay is presented, with the error bars indicating the standard deviation. It can be observed that our model exhibits lower one-way delay compared to GCC. We attribute the higher one-way delay in GCC to its utilization of a high encoding bitrate even in scenarios with limited cellular network available bandwidth. This leads to increased delay when video streaming.

Comparison of broken images severity. To quantilize broken images severity, we use SSIM to evaluate them. From Fig. 6.7, one can clearly see that the overall SSIM values

doi:10.6342/NTU202301100

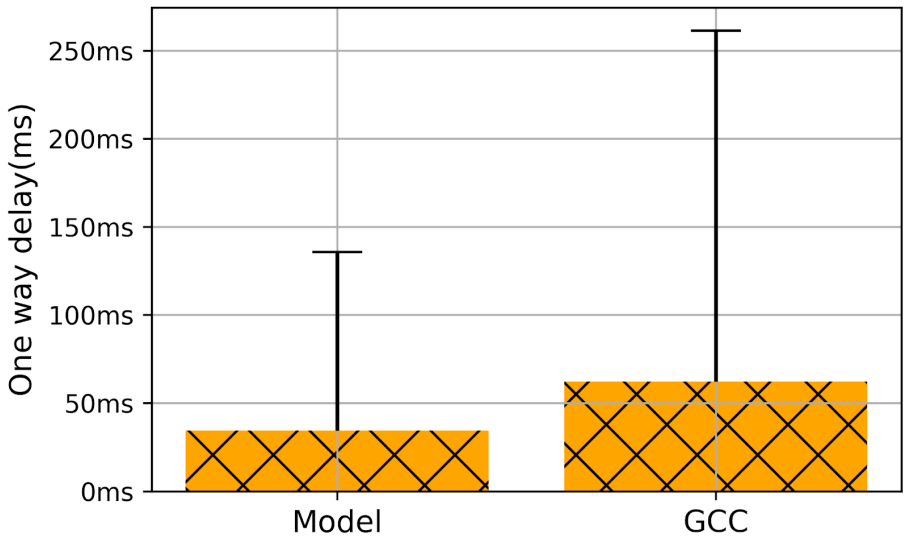


Figure 6.6: Comparison of one way delay.

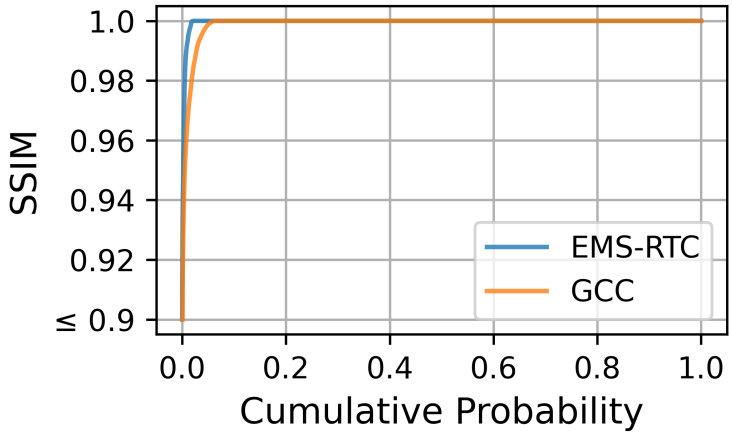


Figure 6.7: Comparison of SSIM CDF.

of our EMS-RTC model are higher than those of GCC. It means that in GCC the number of broken images is higher than EMS-RTC.

Comparison of bitrate utilization. Fig. 6.8 compares the CDF of the applied bitrate when using GCC and EMS-RTC. GCC tends to use a higher bitrate, with the average bitrate of EMS-RTC at about 4554 Kbps and that of GCC at 6126 Kbps. This is in line with our design, which trades the video quality off for a reduced number of stalls.

On the other hand, the difference in image quality in SSIM when using these two rate adaption mechanisms is not significant. To validate this argument, we encode our

doi:10.6342/NTU202301100

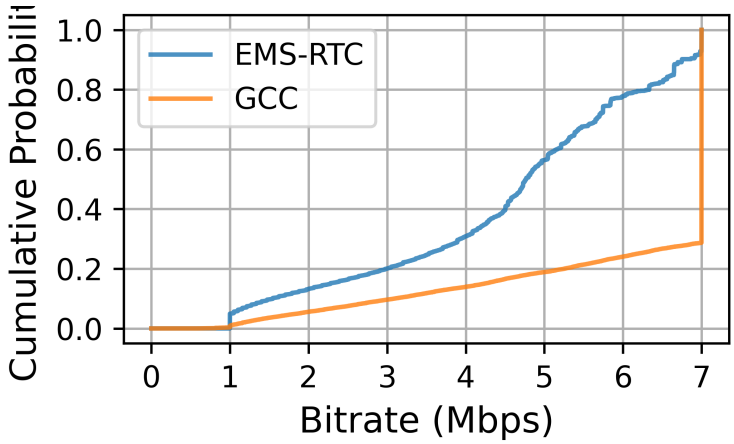


Figure 6.8: Comparison of bitrate CDF.

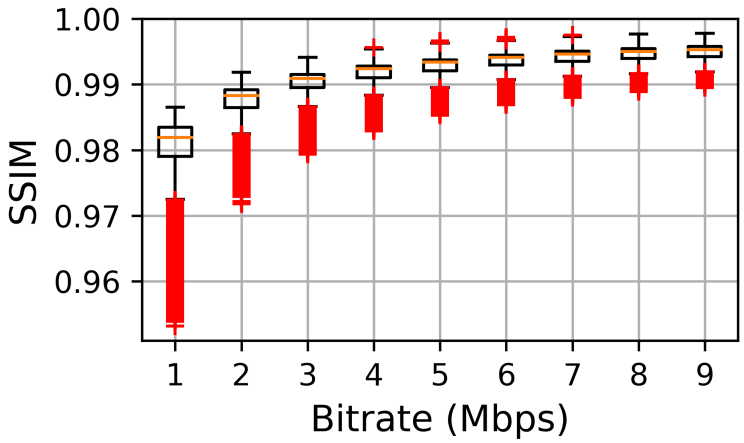


Figure 6.9: Comparison of SSIM with different bitrates.

videos with different bitrates and calculate their SSIM with respect to the original video. As shown in Fig. 6.9, one can see that the reduction of SSIM is no longer significant when the bitrate is larger than 3000 Kbps, which is the usual case for both GCC and EMS-RTC. Therefore, we conclude that for a 1080p video, EMS-RTC can achieve better fluency with little or neglectable video quality reduction. We also compared the SSIM calculated from the received live-stream videos with respect to the original encoded videos at the sender.

doi:10.6342/NTU202301100

Chapter 7 Conclusion

This paper introduces EMS-RTC, a real-time video streaming platform customized for the need for smart ambulance services. EMS-RTC treats bitrate adaptation as a classification problem: increase, decrease, or hold the transmission bitrate to cope with the network condition in the near future. The decisions are made based on a number of parameters from the feedback messages in WebRTC and the received signal power RSRP. We implemented a video streaming platform with the sender on a test car to emulate the operation of streaming videos from an ambulance. This allows us to collect real-world data to train the bitrate adaption model. Based on our user study, we optimize our LSTM-based classifier model to minimize the occurrence of video stalls, with more emphasis to mitigate packet delay and jitter. Real-world evaluation results show that EMS-RTC is able to reduce the number of stall events by a factor of 3.85, compared to GCC.

doi:10.6342/NTU202301100

Chapter 8 Future work

In future work, we have several directions to explore.

1. Despite EMS-RTC can accurately adjust the bitrate, EMS-RTC sometimes may encounter a situation where the base station is overloaded with numerous users connecting, resulting in insufficient available bandwidth. This can cause EMS-RTC to reduce the bitrate, making images blurry. If we collaborate with ISPs, EMS-RTC can obtain first priority of connection, or even obtain dedicated channels specifically for transmitting video streaming of ambulance service.

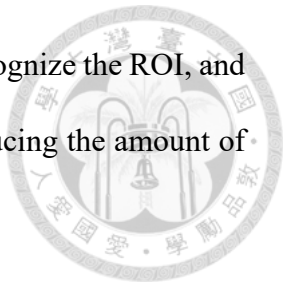
2. A possible future work is to discuss with paramedics to determine the duration of imperceptible stalls. We can lower the criteria for decreasing the bitrate and allow imperceptible stall events to occur. Although EMS-RTC may have some imperceptible stall events, it can achieve better image quality in video streaming.

3. EMS-RTC currently only considers a fixed resolution. In order to enhance QoE, we can also incorporate other adjustable parameters, such as resolution and frame rate, to achieve greater flexibility and customization.

4. To determine the patient's symptoms, doctors may only be interested in the region of the patient in the video streaming. By using ROI encoding, we can prioritize encoding the region of interest (ROI) to make it clearer. We can ask the doctor to

doi:10.6342/NTU202301100

mark the ROI, or even pretrain a machine learning model to recognize the ROI, and only transmit the ROI during the video streaming, thereby reducing the amount of data transmitted.



doi:10.6342/NTU202301100

References

[1] Yunkai Zhai, Xing Xu, Baozhan Chen, Huimin Lu, Yichuan Wang, Shuyang Li, Xiaobing Shi, Wenchao Wang, Lanlan Shang, and Jie Zhao. 5G-network-enabled smart ambulance: Architecture, application, and evaluation. IEEE Network, 35(1):190–196, 2021.

[2] Jinsung Lee, Sungyong Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Hyoyoung Lim, Jihoon Lee, Xiaoqing Zhu, Sangeeta Ramakrishnan, Dirk Grunwald, Kyunghan Lee, and Sangtae Ha. Perceive: Deep learning-based cellular uplink prediction using real-time scheduling patterns. In Proc. ACM International Conference on Mobile Systems, Applications, and Services (MobiSys), page 377–390, 2020.

[3] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and design of the Google congestion control for web real-time communication (WebRTC). In Proc. International Conference on Multimedia Systems, 2016.

[4] WebRTC home. https://webrtc.org/, 2023.

[5] Pion WebRTC: A pure Go implementation of the WebRTC API. https://github.com/pion/webrtc, 2022.

[6] Stefan Holmer, Magnus Flodman, and Erik Sprang. RTP Extensions for Transport-

doi:10.6342/NTU202301100

wide Congestion Control. Technical Report draft-holmer-rmcat-transport-wide-cc-extensions-01, 2015.

[7] Bo-Wen Wang, Wen-Hsuan Shen, Ming-Ju Hsieh, and Hsin-Mu Tsai. Ems-rtc: Lstm-based adaptive video streaming for smart ambulance. In 2023 IEEE Vehicular Networking Conference (VNC), pages 9–16, 2023.

[8] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. Real-time bandwidth prediction and rate adaptation for video calls over cellular networks. In Proc. International Conference on Multimedia Systems (MMSys), 2016.

[9] Balázs Kreith, Varun Singh, and Jörg Ott. Fractal: Fec-based rate control for rtp. In Proceedings of the 25th ACM International Conference on Multimedia, MM '17, page 1363–1371, New York, NY, USA, 2017. Association for Computing Machinery.

[10] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In Proceedings of the conference of the ACM special interest group on data communication, pages 197–210, 2017.

[11] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. Learning to coordinate video codec with transport protocol for mobile video telephony. In Proc. ACM Annual International Conference on Mobile Computing and Networking (MobiCom), 2019.

[12] Huanhuan Zhang, Anfu Zhou, Jiamin Lu, Ruoxuan Ma, Yuhan Hu, Cong Li, Xinyu Zhang, Huadong Ma, and Xiaojiang Chen. OnRL: Improving mobile video telephony via online reinforcement learning. In Proc. ACM Annual International Conference on Mobile Computing and Networking (MobiCom), 2020.

doi:10.6342/NTU202301100

[13] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Alexander Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In NSDI, volume 20, pages 495–511, 2020.

[14] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei. Linkforecast: Cellular link bandwidth prediction in lte networks. IEEE Transactions on Mobile Computing, 17(07):1582–1594, jul 2018.

[15] Qualcomm extensible diagnostic monitor. http://www.qualcomm.com/media/documents/tags/qxdm, 2008.

[16] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, pages 202–215, 2016.

[17] MobileInsight. running your customized plugin on the phone. http://www.mobileinsight.net/tutorial-plugin.html, 2018.

[18] ITU Telecom et al. Advanced video coding for generic audiovisual services. ITU-T Recommendation H. 264, 2003.

[19] Dan Grois, Detlev Marpe, Amit Mulayoff, Benaya Itzhaky, and Ofer Hadar. Performance comparison of h.265/mpeg-hevc, vp9, and h.264/mpeg-avc encoders. In 2013 Picture Coding Symposium (PCS), pages 394–397, 2013.

[20] Bartjan Jansen, Timothy Goodwin, Varun Gupta, Fernando A. Kuipers, and Gil Zussman. Performance evaluation of webrtc-based video conferencing. SIGMETRICS Perform. Evaluation Rev., 45:56–68, 2018.

doi:10.6342/NTU202301100

[21] Telecommunication Standardization Sector of ITU. ITU-T Recommendation G.114: Transmission Systems and Media : General Recommendations on the Transmission Quality for an Entire International Telephone Connection : One-Way Transmission Time. International Telecommunication Union, 2003.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, nov 1997.

[23] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004.

[24] Leo Breiman. Random forests. Machine learning, 45:5–32, 2001.

[25] Yihua Guo, Feng Qian, Qi Alfred Chen, Zhuoqing Morley Mao, and Subhabrata Sen. Understanding on-device bufferbloat for cellular upload. In Proc. Internet Measurement Conference (IMC), page 303–317, 2016.

doi:10.6342/NTU202301100