Noname manuscript No.
(will be inserted by the editor)

UDAB-ViS: User Driven Adaptable Bandwidth Video System

Dustin Wright · Yusuf Ozturk

Received: date / Accepted: date

Abstract Adaptive bitrate (ABR) streaming has become an important and prevalent feature in many multimedia delivery systems, with content providers such as Netflix and Amazon using ABR streaming to increase bandwidth efficiency and provide the maximum user experience when channel conditions are not ideal. Where such systems could see improvement is in the delivery of live video with a closed loop cognitive control of video encoding. In this research, we present a camera system which provides spatially and temporally adaptive video streams, learning the users preferences in order to make intelligent scaling decisions. The system employs a hardware H.264/AVC encoder for video compression. The encoding parameters can be configured by the user or by the cognitive system on behalf of the user when the bandwidth changes. The cognitive video client developed in this study learns the users preferences (i.e. video size over frame rate) over time and intelligently adapts encoding parameters when the channel conditions change. It has been demonstrated that the cognitive decision system developed has the ability to control video bandwidth by altering the spatial and temporal resolution, as well as the ability to make scaling decisions.

Keywords Multimedia Communication · Cognitive Computing · Bandwidth Adaptation · Machine Learning · User Preferences

D. Wright
San Diego State University, Electrical and Computer Engineering Department, 5500 Campanile Dr. San Diego, California 92182
E-mail: wright21@rohan.sdsu.edu

Y. Ozturk
San Diego State University, Electrical and Computer Engineering Department, 5500 Campanile Dr. San Diego, California 92182

1 Introduction

As wireless networks become more ubiquitous and the number of devices capable of accessing these networks increases, the need for more efficient video streaming solutions becomes vastly important. By 2015, it is expected that approximately 90 percent of online consumer traffic and almost 66 percent of mobile traffic will be video [8]. With an increasing amount of video traffic, bandwidth efficiency becomes a serious concern in order to deliver the best quality of experience (QoE) to each individual user. At the same time, servers should be able to deliver video in such a way that information the user deems important is not lost due to bandwidth constraints and the method by which the video adapts.

The rest of the paper is structured as follows; in section 3 we review H.264 packetization. In section 3.2, we review H.264/SVC as a scalable streaming solution, as well as the scalability solution used in our system. Sections 5 and **??** will describe the system architecture proposed in this study, as well as the method by which the video is encoded. Section 4 will detail our learning model and how video bandwidth is optimized. Finally, test setup and experimental results will be presented in section 6, and we conclude in section 7.

2 Related Works

The problem of sending a continous video stream over an uncertain channel is not new and a range of solutions have been proposed. Many of the approaches are based on already existing protocols such as the Real-Time Streaming Protocol RTSP [11], TCP, and HTTP [8,**?**,**?**,**?**] and achieve bandwidth adaptability in one of a few ways. These bandwidth adaptability techniques include progressive download, adaptive bitrate (ABR) streaming, and stream-switching [4]. In progressive download, video is transmitted as regular data files using TCP and is buffered by the client; playing starts when a sufficient amount of buffering has been achieved. With ABR, the server selects the encoding bitrate in order to optimize the video's SNR resolution for a given channel. The result is a drastic reduction in the need for buffering; from the end user's perspective, the quality resolution of the video changes as network conditions change. Examples of ABR solutions are found in HTTP Adaptive Streaming (HAS) [8] and Dynamic Adaptive Streaming over HTTP (DASH) [6], used in the popular video content provider Netflix [2]. Finally, with stream switching, the server encodes the source video with different encoding parameters and allows the client to switch between streams based on network conditions. Examples of stream switching solutions can be found in [4],[1] and [19]. Such techniques are ideal for online video streaming as they can use HTTP to negotiate streaming parameters and transmit the video stream; however, the major pitfall is that in most cases, only the video bitrate will be affected and no control is exercised over the spatial and temporal resolution of the video. In the case where spatial and temporal resolution can be affected, raw video will have to be re-encoded or transcoded at the source which can cause a delay in the video being transmitted [4].

In addition to protocol based approaches which tend to only allow for bitrate scalability, many codec based approaches have been developed which allow for easy spatial and temporal scalability [7]. Two prime examples of codec based approaches are H.264/AVC and H.264/SVC [12][14]. With these codecs, video need only be encoded once and, due to the nature of the decoders, can be transmitted as several sub-streams in order to control the temporal or spatial resolution. The primary issue with this approach is that it limits the client to only a certain set of video decoders. A review of the H.264/SVC approach to scalability will be presented later in this paper.

A recent area of interest seen in the literature is the autmoation of video encoding parameter selection as a means to provide both exceptional quality of service (QoS) and quality of experience (QoE). Approaches to this problem include both cognitive and non-cognitive solutions, though fewer cognitive systems have yet been proposed. In [4], the authors propose a *Quality Adaptation Controller* which uses a proportional-integral (PI) controller at the server to select an appropriate video stream for the client. The system uses stream switching and employs feedback control to maintain the video bandwidth below the available channel bandwidth. Examples of cognitive solutions are found in [10] and [15]. In [10], the authors use statistical models that adapt to user feedback in order to make encoding parameter selections on their behalf.

The authors of [15] propose a multidimensional adaptation (MDA) solution which uses the adaptation-resource-utility (A-R-U) model as a means to intelligently scale a video sequence. Their solution looks at a subjective quality evaluation and clustering in order to classify video sequences. This involves an offline training phase where metadata, video features, and domain specific knowlegde are extracted in order to cluster and label groups of video sequences, as well as an online prediction phase in which adaptation operations are applied to streamed video. The optimal adaptation operations are selected based on the class of the sequence in order to maximize the user experience while staying within the limits of constrained resources (i.e. network capacity).

This study investigates a cognitive approach to video bandwidth control based on user preferences learned by C support vector machines (SVM). This has an advantage over previous cognitive solutions in that it takes into account multiple features related to the video, such as video content, and is scalable in that more features may be observed if needed. We present a solution that is novel in its application of machine learning as an accurate method to learn user preferences.

3 H.264 Relevant Background and Proposed Scaling Architecture

3.1 H.264/AVC Basics

H.264/AVC is a video coding standard developed jointly by the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Pictures Expert Group (MPEG) designed with the goals of enhanced video compression and "network friendly" video representation addressing "conversational" applications such as video conferencing, as well as "non-conversational" applications such as broadcast streaming [18]. In December of 2001 VCEG and MPEG formed a Joint Video Team (JVT) which in March of 2003 finalized the draft of the H.264/AVC video coding standard for formal submission [18].

The standard provides highly efficient video coding andis used in a breadth of applications, from storage to streaming. It provides bitrate savings of 50% or more over its predecessor video codecs [17], making H.264/AVC especially applicable in wireless environments. The effectiveness of H.264/AVC as a tool for video compression over IP networks and in wireless settings is reviewed in [16] and [13]. In addition, H.264 employs a litany of features to enhance the quality of video coding and error resiliency over previous standards. [18] and [5] provide a detailed overview of these features. The basic structure of an H.264/AVC encoder is depicted in Figure 1.

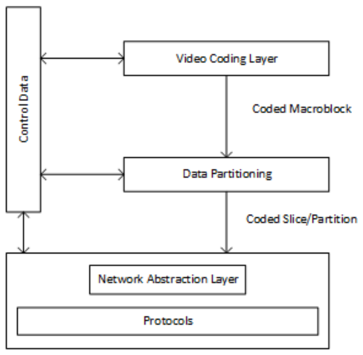

Fig. 1 H.264/AVC Encoder Structure [18]

The codec given in Figure 1 covers both the Video Coding Layer (VCL) and the Network Abstraction Layer (NAL). The VCL performs the physical encoding and compression of video while the NAL wraps a header around video packet data in order to assist the decoder in understanding how to handle the packetized frames. We will now discuss the NAL and give a general overview of how frames are packetized when sent using UDP/RTP.

The NAL allows the ability to map and packetize data with a multitude of transport layer protocols (i.e. UDP/RTP, file formats, etc.). When frames are encoded in the VCL they are organized into NAL units which serve as a wrapper to the underlying information. Each NAL unit contains a header byte that indicates what type of data is contained in this unit. This allows for the segmentation of video into packets, with the NAL unit indicating the start of a new access unit. The NAL unit contains a one byte header and a payload byte string [17]. The header indicates the type of NAL unit, potential presence of errors, and information about the relative importance of this NAL unit in the decoding process [17]. The structure of the one-byte NAL unit header is shown in Figure 2.

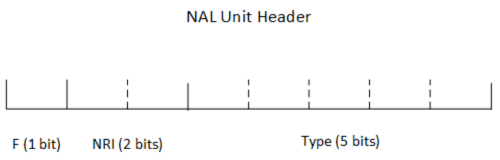

Fig. 2 NAL Unit Header Structure

The fields of the header are designated as follows:

- F: forbidden bit; should always be 0
- NRI: used to indicate if the content of this NAL unit should be used to reconstruct reference pictures in inter picture prediction
- Type: Specifies the NAL unit payload type

Examples of NAL unit types are parameter sets that contain relevant information about a the video stream or an individual frame, as well as frame type (I frame, P frame, B frame) and slice priority.

The H.264/AVC specification also defines a set of profiles and levels which specify different sets of required functional support for decoders. According to [18], "A profile defines a set of coding tools or algorithms that can be used in generating a conforming bit-stream, whereas a level places constraints on certain key parameters of the bitstream." For video conferencing applications or streaming from mobile devices such as the system proposed in this paper, the constrained baseline or baseline profile are appropriate choices. More detailed information on profile types and their constraints can be found in section A.2 of [5].

In the next section, we comment on the scalable video coding extension of H.264 and compare it with the cognitive parameter adaptation method propsed in this paper.

3.2 Video Scaling Architecture

The need for scalable video codecs can be characterized by the following scenario: when transmitting a stream at a certain quality with a video bandwidth B over a congested channel where the channel bandwidth C fluctuates such that $C < B$, the receiving terminal may experience significant degradation of video quality. Scalable codecs combat this by altering one or more resolutions of the video in order to fit the channel. H.264/SVC is once such codec that uses composable bit streams as a means to scale video. Certain parts of the bit stream are removed, separating one stream into layered substreams in such a way that the underlying streams are still decodable [12]. For example, a transmitter may send one base layer bit stream and multiple enhancement layer bit streams with the receiver selecting which of these to send to the decoder. In this, video bandwidth can be controlled by choosing only the necessary bit streams to stay within the channel bandwidth.

SVC presents a sharp contrast to classic single layer video streams in which one decodable bit stream is transmitted. In order to have control over the bandwidth of the video, the transmitter must encode the source video with different encoding parameters and the receiving decoder must adapt to these changes. We will next summarize the Scalable Video Coding extension of H.264 as described in [12] and compare it to the scaling method developed in this study that used a single layer video stream, altering encoding parameters at the source.

The Scalable Video Coding extension of H.264/AVC inherits all of the base functionality of H.264 with only the necessary added features to achieve scalable video streaming. In this, it supports the primary scalability parameters, being temporal, spatial, and quality resolution. To achieve temporal scalability, the transmitter may send multiple temporal streams divided into a temporal base layer and one or more temporal enhancement layers [12]. One may label these streams as T_0 through T_k. A receiving decoder then simply needs to know which of these access units are valid or invalid for the current stream, starting from 0 through n where $n \leq k$. The ability to partition a stream as such and play only the valid streams is already present in the H.264/AVC standard with the employment of reference picture memory control [12]. The partitioning of a stream into multiple temporal streams is illustrated in Figure 3.

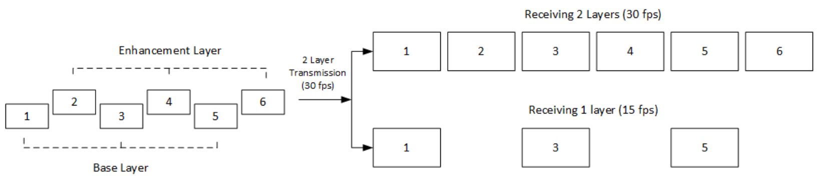
Enhancement Layer
Base Layer
1
2
3
4
5
6
2 Layer Transmission (30 fps)
Receiving 2 Layers (30 fps)
Receiving 1 layer (15 fps)

Fig. 3 Temporal Scalability with H.264/SVC [12]

In order to achieve spatial scalability, SVC uses multi-layer coding with inter-layer prediction [12]. Multiple layers are transmitted, each corresponding to a specific spatial resolution and referred to by an integer valued dependency identifier between 0 and $d - 1$where d is the number of spatial layers [12].

Quality scalability works on the same principle as spatial scalability with the layers transmitted being of the same spatial resolution.

In our proposed single-layer coded video stream, one bit stream is encoded and sent to the receiver. This bit stream is of a fixed spatial, temporal, and quality resolution for the entire sequence of video. There is no mechanism inherent to the codec to change the spatial, temporal, or quality resolution midstream. In this, the video bandwidth is controlled by switching encoding parameters at the source, effectively segmenting the video in time. The video stream is partitioned into multiple sequences labeled $T_k, k = 0 \ldots n-1$ where n is the number of segments for the given session and T_k is the time instance when the segment k begins. The partitioning is determined on the fly as a function of the channel bandwidth, where channel bandwidth can be measured with a reasonable degree of accuracy; for example, using a method like DIChirp as laid out in [9]. At each time instance T_k, the transmitter resets the encoding parameters in such a way that the bandwidth of the video is altered to fit to the channel. When this occurs, a new sequence parameter set is introduced into the stream to signal to the decoder the changes to the encoded video. A sample stream is depicted in Figure 4. The changes to the

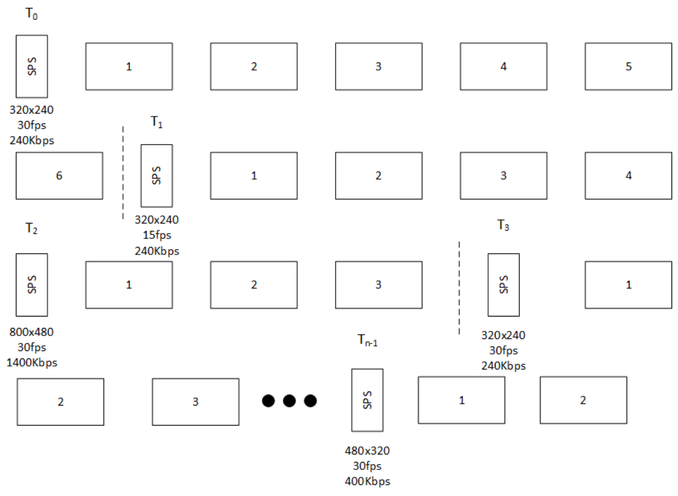

Fig. 4 Single Layer Encoder Parameter Switching

encoding parameters will insert a delay into the stream for the time it takes to restart the encoding process. To compensate for this delay and to handle the alteration events at time T_k, we propose the receiver architecture in Figure 5. In the proposed architecture, a preprocessor inspects the NAL unit headers of each incoming packet, waiting for a new sequence parameter set. At this event, the idle decoder thread is invoked and set up to decode the next sequence of video. The previously active decoder thread empties its queue prior to the start of the new segment of video. When the active thread signals completion,

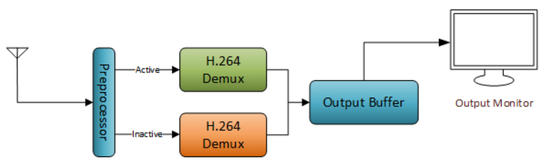

Fig. 5 Receiver Architecture

the new thread takes over. This effectively mitigates any delay that may be introduced due to reconfiguring the decoder, providing a smooth stream for the user.

Delay on the encoder side is greatly avoided by using a hardware H.264/AVC encoder. Encoder initialization happens in real time and there is no CPU overhead when it comes to encoding frames. In addition, this real-time efficiency allows for minimal delay between video segments. Finally, our method allows us to control the temporal, spatial, and quality resolution of each video segment in a more granular fashion than SVC as we are not restricted to a discrete number of video layers.

4 Video Parameter Adaptation

With our developed video scaling architecture, the need arises to be able to intelligently select encoding parameters for the user. When given many options, different users will have different needs when it comes to how video is presented. For example, in the context of viewing sports, a person may require that the frame rate and quality be kept high at all times, while in a medical consultation setting the frame rate may be negligible compared to spatial resolution. In this, we have investigated and present here a scheme for cognitive video parameter selection based on learned user preferences. The proposed learning and prediction mechanism offers a simple, yet effective way to provide dynamically adaptive video streams tailored to each individual.

4.1 Learning User Preferences

In our cognitive system, user peferences are learned using support vector machines (SVMs). A support vector machine is a commonly used tool in supervised learning which has applications in both classification and regression problems. We adopted the LibSVM implementation of a C support vector machine with a radial basis kernel [3] which we utilize in user classification.

Classification starts with a set of training data consisting of a multidimensional set of features and associated known ground truths or labels. Support vector machines function by finding the optimal separation between clusters of training data, where the clusters are differentiated by their label. The result of this is a function used to predict the ground truth for new inputs. The SVM algorithm creates the separation by inserting a hyperplane between the border points of the clusters in the training set, known as the support vectors, and maximizing the margin between these support vectors. Nonlinear separations are found with the use of a kernel: a function which can map the feature space into higher dimensions, depending on the kernel type. Two C support vector machines with radial basis kernels are employed in this system to learn user preferences. For convenience, the C support vector machine implementation defined in [3] is presented and briefly summarized below.

A training set is defined as a feature vector $x_i \in \mathbf{R^n}, i = 1 \ldots l$ and the class label vector $y_i \in \{1, -1\}$ where 1 and -1 indicate the two distinct classes. The primal optimization problem in equation (1) is then solved

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \ldots, l \end{aligned} \tag{1}$$

where $\phi(x_i)$ maps x_i into higher dimensional space and $C > 0$ is a configurable parameter [3].The dual problem presented in equation (2) can then be solved in order to account for the possible high dimensionality of the vector parameter w.

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\ \text{subject to} \quad & y^T \alpha = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, \ldots, l \end{aligned} \tag{2}$$

where e is a column vector of all ones, $Q_(i,j) = y_i y_j K(x_i, x_j)$ and $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is the radial basis function (RBF). The RBF K is a Gaussian distribution, presented in equation (3).

$$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2} \tag{3}$$

where γ is a configurable parameter selected by the user. Finally, the optimal w is computed using equation (4), and the decision function is laid out in equation (5).

$$w = \sum_{i=1}^{l} y_i \alpha_i \phi(x_i) \tag{4}$$

$$sgn(w^T \phi(x) + b) = sgn(\sum_{i=1}^{l} y_i \alpha_i K(x_i, x_j) + b) \tag{5}$$

The final decision function is what our system uses in order to make predictions, and our primary controls over the accuracy of this function are the tuning parameters C and γ.

In order to use the SVMs, we must first develop a pool of training data for a given user. When a new user begins interacting with the system, they are initiated in “learning mode.” In learning mode, a change in channel bandwidth will result in a knee-jerk reaction by the system to simply alter the quality resolution of the video, leaving spatial and temporal resolution unchanged. Explicit feedback is obtained from the user by asking if the video quality is acceptable. A training sample is then recorded and added to the SVM training data. For experimental purposes, we collect at least 30 samples from the user before training. When the training set is sufficiently large enough it is split in half, with one half used for training and one half used for testing. We iterate through different values of the SVM tuning parameters C and γ, training the support vector machines and checking the prediction accuracy using the testing data on each iteration. The final values of C and γ are then selected from the most accurate test. Finally, the decision function is created using our selection of tuning parameters and is used to make class predictions. An overview of the process is shown in Figure 6.

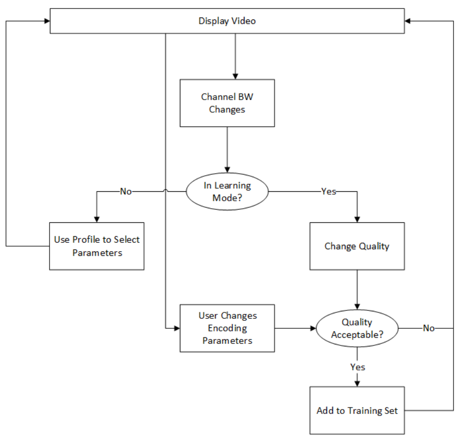

Fig. 6 Feedback Process

Next, we discuss our application of classification in multidimensional video parameter adaption.

4.2 Multidimensional Video Adaptation

The perceived QoE of a video stream is highly dependent on how the video is adapted to changes in network capacity. Multidimensional video adaptation, in which the temporal, spatial, and quality resolution are subject to alteration, is essential for serving various users on disparate devices with individual preferences. To achieve multidimensional adaptation, we combine learned user preferences with our video scaling architecture in order to make intelligent adaptation decisions for individual users.

In our cognitive adaptation system, we estimate the link capacity to determine when to alter the video parameters. The multidimensional adaptation operation that we perform is dependent on a class label predicted at the time of adaptation. In order to predict this label, we use two support vector machines which learn the users perception of the importance of frame rate, frame size, and SNR resolution under different contexts. The context is determined based on features of the video and the streaming environment, such as network bandwidth and video content type. The predicted label takes on values from 0 to 3, conveying the following information:

– Label 0: User weighs both frame rate and frame size with equal importance, while quality is of less importance
– Label 1: User weighs frame rate with highest importance, quality with secondary importance and size with least importance
– Label 2: User weighs frame size with highest importance, quality with secondary importance and frame rate with least importance
– Label 3: User weighs quality with highest importance, and frame size and rate with less importance

The motivation for using these four labels is that we only desire to know the preference of the user for each video dimension. This preference gives us a basis for which to optimize the encoding parameters in such a way that the QoE is maximized. In this, our system performs one of four operations to preferentially alter the encoding parameters.

When developing the pool of training data. our system assigns labels based on the quality, spatial, and temporal resolution of the video when a training sample is recorded. For the purposes of this study we employ a heuristic in which these resolutions are inspected and the ground truth is found by comparing them to a threshold value. For example, suppose the maximum frame size is CIF (352x288) and the maximum frame rate is 30fps. We can define the threshold to be half the max resolution value. In this, if the user is viewing video at QCIF frame size and 30fps, they are given label 1 as defined above. In the same way, if they are viewing video at CIF frame size and at 15fps, they will be given label 2. The remaining labels are determined in the same fashion.

The multidimensional adaptation problem is characterized in [15] by the following:

$$\begin{aligned} \tilde{a} &= \underset{a \in A}{\operatorname{argmax}} \quad U(a) \\ R(a) &\leq R_0 \end{aligned} \tag{6}$$

where a is a multidimensional adaptation function defined in the space $\mathbf{A}$. This model states that the correct adaptation function to choose should maximize the utility $U(a)$, which can be any metric that represents the quality of the user experience. The constraint R_0 is the available resource, in our case being the bandwidth of the network, and $R(a)$ is the new resource requirement due to the adaptation. We can equally represent the constraint as $R(a) = K * R_0, 0 < K \leq 1$. In addition, we assume that $R(a) = \beta$, the total bitrate of the video after the adaptation operation. Given this, we model quality resolution in bits per pixel as follows:

$$l = \frac{\beta}{w * h * s} = \frac{R(a)}{w * h * s} = \frac{K * R_0}{w * h * s} \tag{7}$$

where $w * h$ is the spatial resolution of the video after adaptation and s is the temporal resolution after adaptation. In the adaptation operation, we select the spatial resolution and temporal resolution from a discrete set of values, then calculate the quality resolution l as a result of these selections. We can call these sets $\mathbf{C}$ and $\mathbf{S}$, where c_i is spatial resolution $i, i = 1 \ldots n$ and s_i is temporal resolution $i, i = 1 \ldots m$. We assume that R_0 can be determined within a reasonable degree of accuracy. We then select a value for K, representing the percentage of available bandwidth that is acceptable to fill and providing a bit budget that we use to optimize the video encoding parameters. For this study we also define a value l_{target}, being the minimum target bits per pixel. This value is selected heuristically, but it can theoretically be configured or learned, representing the weight given to quality resolution. To perform the correct adaptation operation, we use the class label predicted by our support vector machines. The operations are defined below.

- a_0: Select c_n and s_m, l is calculated using equation 7
- a_1: Select $s = s_m$ and $l = l_{target}$; maximize c_k such that $w_k * h_k * l \leq \frac{\beta}{s}$. If none exists, select c_0 and calculate l using equation 7
- a_2: Select $c = c_n$ and $l = l_{target}$; maximize s_k such that $s_k * l \leq \frac{\beta}{c}$. If none exists, select s_0 and calculate l using equation 7
- a_3: Minimize c and s, calculate l such that $l = \frac{\beta}{c*s}$

These operations provide an effective way to maximize the QoE based on predicted user classes. Each video parameter is maximized according to the preference of the user, giving precedence to the parameter found most important. In our performance evaluation, we will compare the above method against single dimensional SNR resolution adaptation.

5 System Architecture

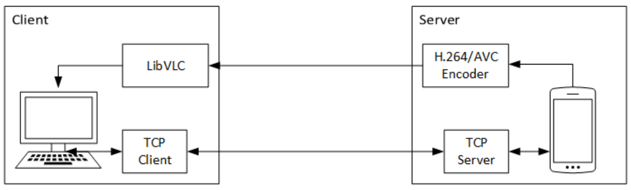

Fig. 7 System Architecture

A basic outline of our system architecture is depicted in Figure 7, with the streaming clients, streaming server, and constituent components. The camera system developed in this study consists of a video client and a streaming server. The client connects to the server to request a new streaming session, at which time live video transmission begins. The streaming server is an Android application implemented on Qualcomm MSM8960 hardware. Video is encoded on the device using a hardware H.264/AVC encoder and streamed to the client using raw UDP packets. The client and server designs are detailed in the following sections.

5.1 Client Design

The client uses LibVLC, a library used in the popular VLC media player developed by the VideoLAN group. LibVLC is equipped with all major functionality required to play our live video streams and handles the decoding of H.264 frames. Through configuration of the media player back end, we set up two decoder threads that are used to demultiplex an H.264 stream, consistent with the architecture described in section **??**. This enables the dynamic change of encoding parameters midstream without seeing a significant effect on video playback.

On top of the lower level streaming layer is a signaling layer utilizing TCP for communication between the server and client. When a new video stream is requested, the client attempts to this TCP connection with the server, and upon successful connection, starts the media player. This client then follows a simple protocol to interact with the server which will be discussed in section 5.3.

The client is responsible for determining the correct choice of encoding parameters and for managing the bandwidth of the video based on the current network capacity. Encoding parameter decisions are based on the user's preferences using the proposed learning model. If an accurate model of the users preferences has been developed, this model drives the parameter selection. A default decision function is employed when learning the users preferences. The client is also intelligent enough not to interfere with the user when they make their own decisions about how to scale the video.

5.2 Server Design

Our camera server application captures live video from an 8MP camera, at varying spatial and temporal resolutions. A TCP server handles all incoming connections from clients and services any requests. On each connection request, a new thread is forked, creating an interface between the client and server. A handle to the encoder is given to each of these threads to allow them control over the resolutions of the video streams. The handle is encapsulated in an object we call the "encoder activation interface". This object, as the name implies, acts as an interface to the encoder (as well as the camera). Via this interface a consumer may initialize, destroy, and alter an encoder for a certain video stream. This allows the clients full control over the parameters of the video, including the video bandwidth. The server remains agnostic of channel conditions and acts as a slave to the connected clients, reconfiguring the stream as necessary based on the request, because the client application learns the users preferences and therefore makes more intelligent decisions about the encoding parameters.

5.3 Session Management

The system contains a signaling layer using TCP for messaging between the client and server. This layer acts as a session manager. TCP is used for reliable communication of messages between terminals as well as to signal the beginning and end of a streaming session. A streaming session begins once the server accepts a client's connection, and ends when one of the terminals disconnects. When a client wishes to receive a particular stream from a server, it first attempts to make a connection with the server. Upon connection, the server initializes a new thread to service the client's requests. The server thread first starts the encoder, which begins streaming packets containing the encoded H.264 frames. This thread then enters a loop in which it responds to the client's requests until it detects that the client has disconnected. We have defined a very simple protocol for submitting such requests in which the client either requests to alter encoding parameters or stop the video stream. To update the encoder, the client sends the following message:

```
<request action="start">
        <width>#</width>
        <height>#</height>
        <fps>#</fps>
        <rate>#</rate>
</request>
```

where "width" is the new desired width, "height" is the new desired height, "fps" is the new desired frame rate, and "rate" is the new desired video bitrate. To stop the encoder, the following message is sent with the request action as stop:

```
<request action="stop" />
```

Upon disconnecting, the thread processing the client's requests will shut down the encoder, stop the video stream, and exit.

6 Experimental Results

To validate the cognitive video scaling solution, we have developed a test bed as shown in Figure 8.

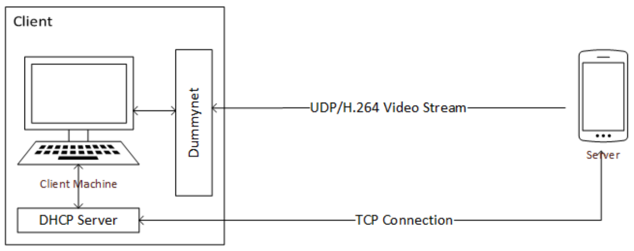

Fig. 8 Camera System Test Bed

We created a point to point connection between the client and server by setting up the client machine as a DHCP server and connecting it directly to the embedded video server, allocating it an IP address on an arbitrary subnet. We are using dummynet , a widely used network emulator, to emulate the behavior of internet in the lab environment. With dummynet, one can control the traffic over a specific channel by limiting bandwidth, inserting packet losses, inserting delay, etc. In order to simulate bandwidth change detection the client simply reads from a file that contains channel bandwidth information. In our test set up we developed a test application which simultaneously sets the bandwidth of the channel to varying values at certain intervals using dummynet, and writes this bandwidth to a flat file that the client can read from. With this, we are able to implement some of the conditions of a real network and be aware of the bandwidth in the client application.

The first experiment tests the ability of the system to control the bandwidth of the video by altering the spatial resolution. Channel bandwidth is kept constant, as well as frame rate which is kept at 30 frames per second. The spatial resolution is changed from 800x480 to 480x320 to 320x240. We used Wireshark to capture and display the bandwidth of the video, obtaining results in Figure 9.

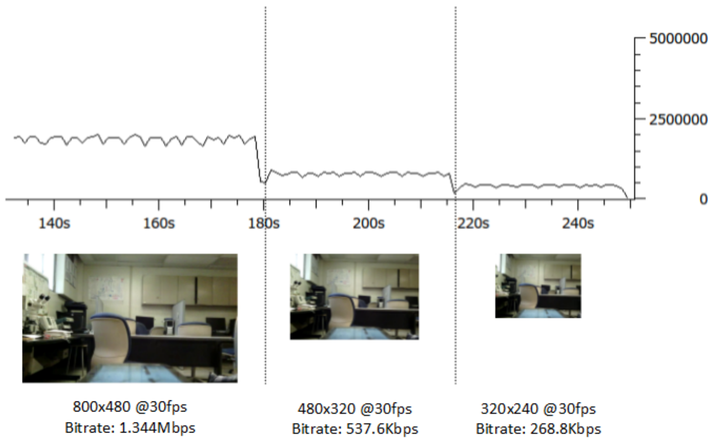

Fig. 9 Spatial Resolution Experiment

As the spatial resolution is changed, the bandwidth of the video changes accordingly. One can easily deduce that this change is directly proportional to the change in resolution. For example, when the spatial resolution changes from 800x480 to 480x320, the total pixel ratio and the ratio of video bandwidth are equivalent:

$$\frac{480 * 320}{800 * 480} = \frac{76800}{384000} = 0.4$$

$$\frac{537.6Kbps}{1344Kbps} = 0.4$$

These results indicate that we have successfully demonstrated control over the video bandwidth by altering the spatial resolution of the video.

In our next test, we showed that we can control video bitrate by altering the amount of compression (number of bits per pixel), resulting in a change in video quality. We tested the systems response to changes in channel bandwidth by altering the bandwidth from 1600Kbps to 800Kbps to 400Kbps. Spatial resolution was kept constant at 800x400 and temporal resolution was kept constant at 30 fps. The resulting changes in video bitrate, as well as playback quality, are depicted in Figure 10.

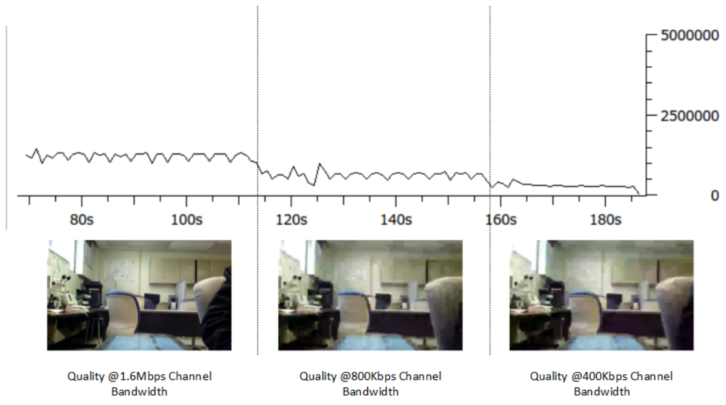

Fig. 10 Quality Resolution Experiment

As can be seen from Figure 15, the system successfully and immediately responds to changes in channel bandwidth by reducing the quality resolution of the video. In addition, the reduction in video bitrate is directly proportional to the reduction in available bandwidth.

Finally, we tested the systems ability to determine the users preferences and scale the video appropriately when channel bandwidth changes. For the purposes of this study, we provided the learning mechanism with an arbitrary training set with the intent to prove the systems ability to properly learn and make predictions. By using a predefined training set we know in advance what classes should be predicted, allowing us to validate the accuracy of the cognitive mechanism by comparing the experimental predicted values with the expected values. The training set used is given in Table 1 and the class distribution for this training set is graphed in Figure 11.

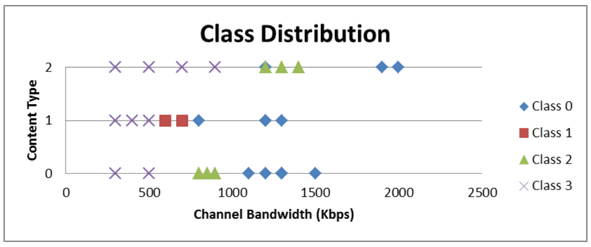

Fig. 11 Class Distribution

The “Content Type” feature is used to classify different scenes that are transmitted in real world scenarios. For example, a content type of 0 may be a video conferencing application with a “talking head” based scene, while a content type of 2 may be a medical based scene that requires extremely high

Table 1 Training Set

Channel Bandwidth (Kbps)	Content Type	Class Label
1200	0	0
1300	0	0
800	0	2
500	0	3
900	0	2
300	0	3
1100	0	0
1300	0	0
1500	0	0
850	0	2
500	1	3
1200	1	0
600	1	1
1300	1	0
300	1	3
400	1	3
800	1	0
600	1	1
700	1	1
1200	2	0
500	2	3
1200	2	0
2000	2	0
1300	2	2
300	2	3
900	2	3
1400	2	2
1900	2	0
700	2	3
1200	2	2

bandwidth and video quality. In this experiment, 3 different content types are used, giving the approximated expected prediction values in Table 2.

Table 2 Expected Prediction Values

Channel Bandwidth (Kbps)	Content Type	Expected Class
0-500	0	3
500-1000	0	2
1000+	0	0
0-500	1	3
500-800	1	0
800+	1	0
0-900	2	3
900-1700	2	2
1700+	2	0

We trained the support vector machines and tested the learning mechanism by setting the bandwidth to 1600Kbps, 800Kbps, and 400Kbps while viewing video streams with content types 0, 1, and 2. For equation (**??**) we selected the

K parameter to be 0.85 and for equation (**??**) we selected the Q_{max} parameter to be 3.5. In all the experiments the output bitrates are selected from equation (**??**). The resulting bandwidth changes, encoding parameter changes, and class predictions are given in Figures 12 - 14.

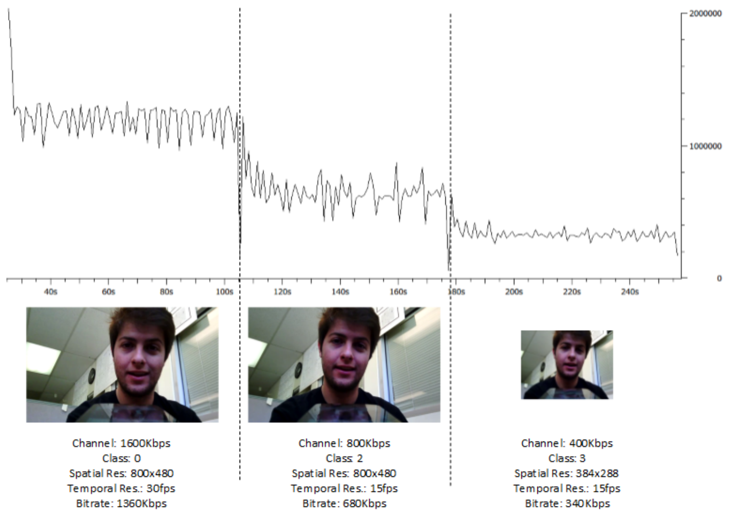

Fig. 12 Predictions With Content Type 0

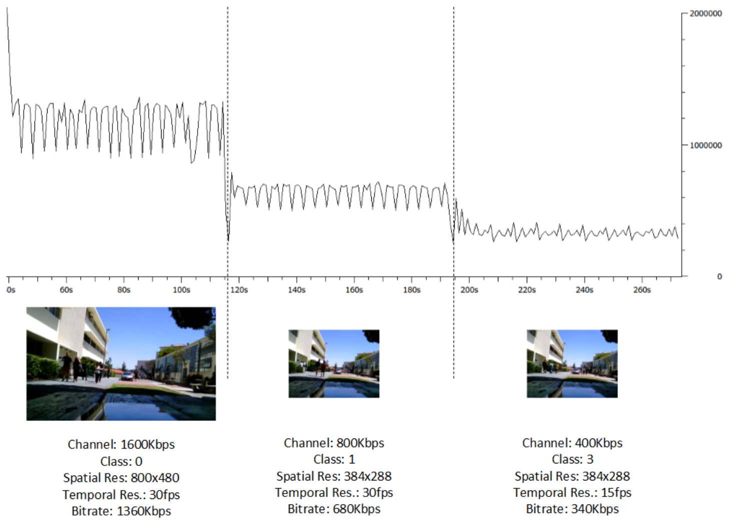

Fig. 13 Predictions With Content Type 1

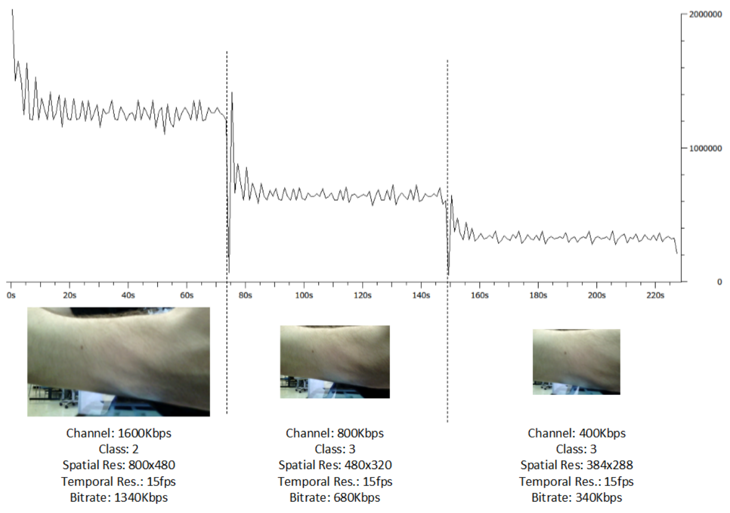

Fig. 14 Predictions With Content Type 2

In all cases, the video bandwidth was adapted when the channel bandwidth changed. In addition, the video bandwidth was consistently kept at 85% of the channel bandwidth as a result of our selection of K. In order to validate the accuracy of the predictions, Table 3 compares the expected class values and the predicted class values at time instances T_0, T_1, and T_2, where T_0 is the instant when the channel bandwidth changes to 1600Kbps.

Table 3 Expected vs. Predicted Classes

Segment	Content Type	Expected Class	Predicted Class
T_0	0	0	0
	1	0	0
	2	3	3
T_1	0	2	2
	1	1	1
	2	3	3
T_2	0	3	3
	1	3	3
	2	3	3

As the table indicates, the support vector machines predicted the correct class with 100% accuracy. In addition, the resulting changes to the encoding parameters followed the changes defined in Figure **??**. When class 0 was predicted, the spatial and temporal resolutions were kept high at the cost of fewer bits per pixel. When class 1 was predicted, the temporal resolution was kept high and the spatial resolution was reduced. The prediction of class 2 resulted in a loss in temporal resolution with the spatial resolution being kept high.

Finally, when class 3 was predicted, the spatial and temporal resolutions were reduced, resulting in greater quality with more bits per pixel.

7 Conclusion

As the volume of internet traffic related to video streaming increases, the importance of having exceptional bitrate adaptation schemes grows. In addition, these schemes should be aware of not only the channel bandwidth, but the surrounding context of the video being streamed. This context encapsulates the content type of the video, and can extend into other dimensions such as amount of motion, geospatial location, and more. We have presented a solution that takes rate adaptation beyond simply changing the quality of the video when the channel bandwidth becomes limited. Our system determines the users preferences about video quality, taking into account if the user prefers a drop in temporal or spatial resolution versus quality resolution. We have demonstrated the ability to adapt to channel bandwidth changes by altering these resolutions. In addition, we have shown that by using support vector machines, we can learn the users preferences and successfully adapt the video bandwidth in line with these preferences. Such a system is a viable solution to the changing atmosphere of video providers, contexts, and the increasing and diverse consumer base.

Acknowledgements

This study in part has been supported by Qualcomm University program.

References

1. Http live streaming overview. Apple Inc. Available from: `https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/StreamingMediaGuide.pdf`
2. Adhikari, V.K., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M., Zhang, Z.L.: Unreeling netflix: Understanding and improving multi-cdn movie delivery. In: 2012 Proceedings IEEE INFOCOM, pp. 1620–1628. IEEE (2012)
3. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) **2**(3), 27 (2011)
4. De Cicco, L., Mascolo, S., Palmisano, V.: Feedback control for adaptive live video streaming. In: Proceedings of the second annual ACM conference on Multimedia systems, pp. 145–156. ACM (2011)
5. ITU-T RECOMMENDATION, H.: 264 advanced video coding for generic audiovisual services. ISO/IEC **14496** (2003)
6. Lohmar, T., Einarsson, T., Frojdh, P., Gabin, F., Kampmann, M.: Dynamic adaptive http streaming of live content. In: 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–8. IEEE (2011)
7. Ohm, J.R.: Advances in scalable video coding. Proceedings of the IEEE **93**(1), 42–56 (2005)
8. Oyman, O., Singh, S.: Quality of experience for http adaptive streaming services. IEEE Communications Magazine **50**(4), 20–27 (2012)

9. Ozturk, Y., Kulkarni, M.: Dichirp: direct injection bandwidth estimation. International Journal of Network Management **18**(5), 377–394 (2008). DOI 10.1002/nem.674. URL `http://dx.doi.org/10.1002/nem.674`
10. Pan, C.H., Lee, I.H., Huang, S.C., Lian, C.J., Chen, L.G.: A quality-of-experience video adaptor for serving scalable video applications. Consumer Electronics, IEEE Transactions on **53**(3), 1130–1137 (2007)
11. Schulzrinne, H.: Real time streaming protocol (rtsp) (1998)
12. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable video coding extension of the h. 264/avc standard. IEEE Transactions on Circuits and Systems for Video Technology **17**(9), 1103–1120 (2007)
13. Stockhammer, T., Hannuksela, M.M., Wiegand, T.: H. 264/avc in wireless environments. IEEE Transactions on Circuits and Systems for Video Technology **13**(7), 657–673 (2003)
14. Unane, I., Urteaga, I., Husemann, R., Del Ser, J., Roseler, V., Rodriguez, A., Sanchez, P.: A tutorial on h.264/svc scalable video coding and its tradeoff between quality, coding, efficiency, and performance. In: J.D.S. Lorente (ed.) Recent Advances on Video Coding. InTech (2011). Available from: `http://www.intechopen.com/books/recent-advances-on-video-coding/`
15. Wang, Y., van der Schaar, M., Chang, S.F., Loui, A.C.: Classification-based multidimensional adaptation prediction for scalable video coding using subjective quality evaluation. Circuits and Systems for Video Technology, IEEE Transactions on **15**(10), 1270–1279 (2005)
16. Wenger, S.: H. 264/avc over ip. IEEE Transactions on Circuits and Systems for Video Technology **13**(7), 645–656 (2003)
17. Wenger, S., Stockhammer, T.: Rtp payload format for h. 264 video (2005)
18. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the h. 264/avc video coding standard. IEEE Transactions on Circuits and Systems for Video Technology **13**(7), 560–576 (2003)
19. Zambelli, A.: Iis smooth streaming technical overview. Microsoft Corporation **3** (2009)