NBS: a network-bandwidth-aware streaming version switcher for mobile streaming applications under fuzzy logic control

Pengcheng Xiong †, Jialie Shen ‡, Qingyang Wang †³, Deepal Jayasinghe †⁴, Jack Li †⁵, Calton Pu †⁶
†College of Computing, Georgia Institute of Technology
266 Ferst Dr., Atlanta, GA 30332, USA
1,3,4,5,6{pxiong3,qywang,deepal,jack.li,calton}@cc.gatech.edu
‡School of Information Systems, Singapore Management University
60 Stamford Rd., Singapore 178900, Singapore
jlshen@smu.edu.sg

Abstract—Good quality of service for a video streaming application under time-varying network bandwidth capacity is often required in many real-world scenarios, e.g., streaming in a typical high-speed vehicular environment. However, developing the enabling algorithms and techniques have been proven to be very challenging. Adapting bitrate streaming method can detect a user’s network bandwidth availability in real time and then adjust the quality of a video stream accordingly. Although this method brings in many advantages, it suffers from an additional storage and encoding costs, and challenges with maintaining quality globally.

We propose NBS (a Network-Bandwidth-aware streaming version Switcher) system for video streaming applications. Compared with adaptive bitrate streaming method, our method is a lightweight one by switching among different versions (e.g., high quality, medium quality, low quality) that a video streaming server provides. Moreover, our method works in an appropriate way that can balance both the responsiveness and the stability. On one hand, responsiveness is necessary because adaptive actions should be taken in a real-time manner to sustain the live streaming when the network bandwidth changes. On the other hand, too sensitive actions will deteriorate the stability and affect the perceptual quality.

NBS is developed based on a feedback fuzzy controller to support the dynamic and adaptive switching. In the presence of the traditional control approaches, the fuzzy logic enables NBS to enjoy much better responsiveness and stability. We have conducted comprehensive experiments to evaluate our approach and the experimental results show that the proposed approach can effectively up-scaling or down-scaling the quality of the stream according to the network bandwidth availability by a smooth and nearly unnoticeable switch without disrupting the continuous playback.

Keywords—Video streaming application, Fuzzy control, Mobile streaming services, Network bandwidth provisioning

I. INTRODUCTION

The introduction of mobile hand-held devices (e.g., smartphones and tablets), the multitude of advances attained in terminal computers, and the deployment of high speed networks have led to a recent surge of multimedia, especially video streaming vendors, e.g., YouTube [1], [2], Netflix [3], PPLive [4], and PPSStream [5]. It is reported that the streaming media business will grow by 27% per year, generating over USD 78 billion in revenue in the U.S. alone by 2014 [6].

Most of the recent video streaming technologies have adopted a nonadaptive approach, wherein the bit-rate and quality of the video are selected prior to the start of the streaming, and fixed during a streaming session. This nonadaptive approach works well for a stable video viewer where a wired connection is usually deployed with high and sufficiently stable bandwidth capacity.

However, this approach can seriously compromise the perceptual quality experienced by the mobile video viewers where bandwidth-related limitations and fluctuations are often found in the mobile access networks, e.g., Wi-Fi, GSM and Wireless Wide Area Networks (WWAN). On one hand, to conservatively choose a low quality version with a highly available network bandwidth is a waste of resources. On the other hand, to aggressively choose a high quality version with a limited network bandwidth will result in either lots of “glitches” or long delay.

A concrete scenario is in the context of vehicular mobility, where a viewer on a moving vehicle watches the video stream using a mobile device (smartphones and tablets) connected to the Internet via a WWAN connection. Due to the heterogenous radio characteristics and WWAN network load conditions at different locations as a vehicle moves along its route, the WWAN bandwidth fluctuates significantly. When the WWAN bandwidth suddenly drops significantly below the selected streaming rate, the video viewer can suffer from noticeable “glitches” caused by frame loss or frequent pauses in playback caused by the exhaustion of the playout buffer [7]. This motivates us to adapt the quality of a video stream according to a user’s bandwidth capacity in real time.

Some providers (e.g., Adobe [8], Apple [9], Microsoft [10] and Octoshape [11]) utilize adaptive bitrate streaming technique [12]. The technique can detect a user’s network bandwidth and CPU capabilities in real time and then adjust the quality of a video stream [13]. This requires the providers to encode a single video at multiple bitrates and switch to the most appropriate one in a real-time manner. Adaptive bitrate streaming brings in many advantages, such
as little buffering, fast start time and a good experience for both high-bandwidth and low-bandwidth connections. For example, Apple will check periodically to see which bitrate chunk it should serve to the viewer when a video is playing. Microsoft will also check every two seconds for the speed of bitrate chunks and pump out the next chunk accordingly. However, the main problem is that HTTP based adaptive bit rate technologies are significantly more operationally complex than traditional streaming technologies. For example, there are currently only two companies making live encoders, i.e., Inlet and Envivio. These encoders are quite expensive. Some of the other documented considerations include additional storage and challenges with maintaining quality globally.

Some providers (e.g., YouTube) take a static approach by maintaining multiple copies of the same video with different quality (i.e., different resolutions) as shown in Fig. 1. It is up to a user to make the adaptation decision of which format to watch based on his/her network bandwidth. The problem with the static approach is that a user does not know much about his/her network bandwidth availability at the first place and it is easy to make a wrong choice. Most of the users have to try several versions back and forth manually until they can find a version that is suitable to play under their network condition.

Classical control theory [14] does not work well for adaptive streaming version switching. A classical controller upgrades to a high quality version when the network bandwidth utilization is below the setting point and downgrades to a low quality version when the network bandwidth utilization is above the setting point. When the available network bandwidth is unfortunately fluctuating between two kinds of bitrates, the versions will be changed back and forth between these two versions. This instability affects the perceptual quality.

Motivated by the concerns about adaptivity and stability above, a fuzzy controller [15], [16] is utilized in our framework due to the following comparisons. (1) We need to build a model to describe the behavior of the system in precise mathematical terms for a traditional proportional or proportional integral controller [14]. The input of the model should be the available video streaming versions and the output of the model should be the real network bandwidth utilization. This model will be difficult to obtain due to the limited video streaming versions and statistical inaccuacries existing in an approximate black-box plant model. However, since a fuzzy logic controller has a model-free nature, we can derive the control strategy following the expert’s control rules. (2) The control knobs, i.e., video version switching can derive the control strategy following the expert’s control rules. (2) The control knobs, i.e., video version switching can derive the control strategy following the expert’s control rules. (3) The control knobs, i.e., video version switching can derive the control strategy following the expert’s control rules.

Based on fuzzy logic, in this paper we propose NBS, i.e., a Network Bandwidth-aware version Switcher for video streaming application. Comparing with the existing approaches, our NBS has two distinguished features. (1) It can help the user to automatically switch among different video streaming versions according to the time-varying network bandwidth availability. (2) It has better perceptual quality with decent responsiveness and stability.

The rest of the paper is organized as follows. Section II outlines the background and the system architecture. In Section III, we show the details for the design of a fuzzy controller. Section IV presents empirical evaluation results of our proposed scheme under a real environment. Related papers are summarized in Section V before Section VI concludes the paper.

II. BACKGROUND AND SYSTEM ARCHITECTURE

In this section, we first discuss the network bandwidth utilization setting that we use in the paper. Then we give a detailed description of the system architecture and the communication among all the components in the architecture that we use throughout the paper.

A. Network bandwidth utilization

Bandwidth is a network QoS parameter that refers to the data rate supported by a network connection or interface. The most common term for bandwidth is bits per second (bps), i.e., effective number of data units transported per unit time. Multimedia applications usually require high bandwidth as compared to other general applications. For example, MPEG2 video requires around 4Mbps, while MPEG1 video requires about 1-2 Mbps. A too high network bandwidth utilization setting like 100% is vulnerable to the network bandwidth availability because there will be package loss once the network bandwidth capacity or the video streaming bitrate fluctuates. A too low network bandwidth utilization setting like 10% will only waste network bandwidth. In our paper, we assume an appropriate network
bandwidth utilization as 80%. Note that, this value strongly depends on the stable property of the network bandwidth capacity and the video streaming bitrate. Under relatively more constant network bandwidth capacity and video streaming bitrate, this value can be set relatively higher, i.e., 90% or more.

B. System architecture

Our system architecture is shown in Figure 3. It is composed of two machines, serving as a video server and a video client (or video user), respectively. The video server emulates YouTube which maintains multiple copies of the same video with different quality. The video client receives the video frames of different versions according to the version switcher from the server machine through the network. Our Network Bandwidth-aware version Switcher (NBS) is deployed with the video client on the user machine. NBS is based on a feedback fuzzy controller which contains a sensor, a controller and an actuator. Firstly, the sensor monitors the network bandwidth usage by the video streaming. Then the network bandwidth utilization is derived by the network bandwidth usage and the network bandwidth capacity. Note that we deliberately change the bandwidth availability to simulate the bandwidth fluctuation. Secondly, the network bandwidth utilization is compared with the network bandwidth utilization setting. The controller makes the decision by specifying the next video streaming version. Finally, the actuator realizes the controller’s decision by switching to the next version.

III. FUZZY CONTROLLER DESIGN

A fuzzy controller adopts formal fuzzy logic control theory to support the desired network bandwidth utilization even when network bandwidth availability varies dynamically as shown in Figure 2.

The fuzzy controller is mainly composed of three parts, i.e., a fuzzifier, a fuzzy inference engine and a defuzzifier. The fuzzy inference engine takes fuzzy sets as input, and produces output in the form of fuzzy sets. Before we describe all the parts, we put the notations that we use in Table I.

<table>
<thead>
<tr>
<th>$k$</th>
<th>control interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{ref}$</td>
<td>network bandwidth utilization setting</td>
</tr>
<tr>
<td>$u(k)$</td>
<td>network bandwidth utilization for the $k$th interval</td>
</tr>
<tr>
<td>$e(k)$</td>
<td>the error of network bandwidth utilization</td>
</tr>
<tr>
<td>$\Delta e(k)$</td>
<td>the change of the error of network bandwidth utilization</td>
</tr>
<tr>
<td>$v(k)$</td>
<td>video version for the $k$th interval</td>
</tr>
<tr>
<td>$\Delta v(k)$</td>
<td>the change of the video version for the $k$th interval</td>
</tr>
</tbody>
</table>

A. Fuzzifier

The fuzzifier is used to describe the inputs to the fuzzy controller by the linguistic variables. The network bandwidth utilization $u(k)$ is measured during the $k$th interval. Then $u(k)$ is compared with $u_{ref}$, i.e., the setting point for the network bandwidth utilization. There are two inputs to the fuzzy controller at the $k$th interval, i.e., the error (i.e., fuzzified $e(k)$) and the change in error (i.e., fuzzified $\Delta e(k)$). They can be calculated as:

$$e(k) = u_{ref} - u(k)$$

and

$$\Delta e(k) = e(k) - e(k - 1)$$
Linguistic variables are associated with linguistic values to describe characteristics of the variables. The candidate linguistic values are positive large (PL), positive medium (PM), positive small (PS), zero (ZE), negative large (NL), negative medium (NM) and negative small (NS). We use $L$ to denote the candidate linguistic value of a variable, e.g., $L_{k}(e) = \text{ZE}$ means that the fuzzification interface converts $e(k)$ to a linguistic value of zero (ZE).

![Figure 4: Membership function](image)

During the conversion, a variable may have multiple candidate linguistic values. We use the membership function in Figure 4 to quantify the certainty of $e(k)$ and $\Delta e(k)$ values to be associated with specific linguistic values. The membership function is designed based on the domain knowledge and expert experience. Specifically, the horizontal axis of Figure 4 represents $e(k)$, $\Delta e(k)$, or $\Delta v(k)$ while the vertical axis indicates the membership value. For membership functions (except for the leftmost or the rightmost ones), we use symmetric triangles of an equal base and 50% overlap with adjacent membership functions. Unlike traditional set theory, the set membership is not binary but continuous to deal with uncertainties in fuzzy set theory. Thus, a fuzzy input or output may belong to up to two adjacent sets in our membership functions with different certainty values. For example, assume that we have $e(k) = 0.083$ and $\Delta e(k) = -0.067$. Then we have $u_{PS}(e(k)) = 0.75$, $u_{PM}(e(k)) = 0.25$ and $u_{NS}(\Delta e(k)) = 1$ following Figure 4. Here $u_{PS}(e(k)) = 0.75$ means that the certainty of $L_{e(k)} = \text{PS}$ is 0.75 as shown in Figure 4.

**B. Fuzzy inference engine**

Based on the fuzzified $e(k)$ and $\Delta e(k)$, the fuzzy inference engine determines the version changes $\Delta v(k)$ to apply at the $(k + 1)$th interval using rule base as below.

$$L_{\Delta v(k)} = \text{Rulebase}(L_{e(k)}, L_{\Delta e(k)})$$

We generate the rule base relying on domain knowledge and experts experience as follows. First, we can derive the linguistic variable $L_{e(k+1)}$ for $e(k + 1)$ in the next interval $k + 1$.

$$L_{e(k+1)} = L_{e(k)} + L_{\Delta e(k)} - L_{\Delta v(k)}$$

Secondly, from the domain knowledge and experts experience, we expect $e(k + 1)$ (the error of network bandwidth utilization for the $(k + 1)$th interval) as close as possible to zero.

$$L_{e(k+1)} = \text{ZE}$$

Combining the previous two formula, we have

$$L_{e(k+1)} = L_{e(k)} + L_{\Delta e(k)} - L_{\Delta v(k)} = \text{ZE}$$

Finally, we can derive that our rule base should satisfy

$$L_{\Delta v(k)} = \text{Rulebase}(L_{e(k)}, L_{\Delta e(k)}) = L_{e(k)} + L_{\Delta e(k)}$$

For example, when $L_{e(k)} = \text{"Positive large"}$ (PL) and $L_{\Delta e(k)} = \text{"Negative small"}$ (NS), we have $L_{\Delta v(k)} = L_{e(k)} + L_{\Delta e(k)} = \text{"Positive medium"}$ (PM). More specifically, when $L_{e(k)}$ is positive large and $L_{\Delta e(k)}$ is negative small, the network bandwidth is far below the reference point and slowly approaches the reference point. We need $L_{\Delta v(k)}$ to be positive so that we could expect network bandwidth to be at the reference point. Positive $L_{\Delta v(k)}$ means that the fuzzy controller will switch to a higher bit rate version which will help increase the network bandwidth utilization. We put (PM) in the cell of row (PL) and column (NS). We fill in all the other cells accordingly in the rule base as shown in Figure 5.

<table>
<thead>
<tr>
<th>$e/\Delta e$</th>
<th>NL</th>
<th>NM</th>
<th>NS</th>
<th>ZE</th>
<th>PS</th>
<th>PM</th>
<th>PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
</tr>
<tr>
<td>NM</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
</tr>
<tr>
<td>NS</td>
<td>NL</td>
<td>NL</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
</tr>
<tr>
<td>ZE</td>
<td>NL</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
</tr>
<tr>
<td>PS</td>
<td>NM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>PM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
</tr>
<tr>
<td>PL</td>
<td>ZE</td>
<td>PS</td>
<td>PM</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
<td>PL</td>
</tr>
</tbody>
</table>

![Figure 5: Fuzzy rulebase](image)

Thus, in the previous example where $u_{PS}(e(k)) = 0.75$, $u_{PM}(e(k)) = 0.25$ and $u_{NS}(\Delta e(k)) = 1$. The IF-THEN rules, rule(PS,NS)=ZE and rule(PM,NS)=PS, in Figure 5 apply. To compute the certainty value of the premise in the corresponding IF premise THEN consequent rule(s), we take the minimum between the certainty values of $e(k)$ and $\Delta e(k)$, because the consequent cannot be more certain than the premise [17]. Thus, $u(PS,NS) = \min(0.75, 1) = 0.75$ and $u(PM,NS) = \min(0.25, 1) = 0.25$. Also, note that maximum four rules apply at a sampling point, since the error or change in error can belong to up to two membership functions in Figure 4. Thus, the worst case time complexity of our fuzzy logic control is $O(1)$. Also, storing the rule-base in Figure 5 consumes a limited amount of memory.
C. Defuzzifier

The defuzzification interface converts the linguistic control signal \( L_{\Delta v(k)} \) to a control signal \( \Delta v(k) \) as follows.

\[
\Delta v(k) = \frac{\sum L_{\Delta v(k)} c(L_{\Delta v(k)}) * u(L_{\Delta v(k)}, L_{\Delta e(k)})}{\sum L_{\Delta v(k)} u(L_{\Delta v(k)}, L_{\Delta e(k)})}
\]

where \( c(L_{\Delta v(k)}) \) means the middle of the triangle’s base. That is, for each combination of \( L_{\Delta v(k)} \) and \( L_{\Delta e(k)} \), we can derive a linguistic value \( L_{\Delta v(k)} \). We defuzzify it through \( c(L_{\Delta v(k)}) \). We finally take the average of all the combinations. If we denote the highest and lowest quality versions as \( HQ \) and \( LQ \) respectively, the adapted video version in the \((k + 1)\)th interval is as follows:

\[
v(k + 1) = \begin{cases} 
HQ & \text{if } v(k) + \Delta v(k) > HQ \\
LQ & \text{if } v(k) + \Delta v(k) < LQ \\
\lfloor v(k) + \Delta v(k) \rfloor & \text{otherwise}
\end{cases}
\]

Thus, in the previous example, since \( u(PS, NS) = 0.75 \) and \( u(PM, NS) = 0.25, \Delta v(k) = (0.75 + 0 + 0.25 \times 0.67) / (0.75 + 0.25) = 0.1675 \).

IV. Evaluation

A. Experimental setup

1) Physical environment and implementation: The two machines in our testbed as shown in Figure 3 are with Intel Core2 Duo 2.4GHz CPU, 4 GB RAM and 1Gb network card. Both of the machines are running Windows XP. We isolate these two machines from any other machines and only monitor the traffic between them.

We use VLC [18] with HTTP-mode as the video server and client. We implement a simple web server, which provides different URLs for different versions of the video stream. We implement a monitor which monitors the bandwidth usage of the user by using the APIs provided by VLC. We implement an actuator by changing the connection URL. In order to simulate the network bandwidth variation, we use Dummynet [19] to control the bandwidth availability. Note that real-time estimation of available bandwidth is a well-known problem for which there are no good, accurate and reliable solutions. We do not monitor the available bandwidth but assume that the available bandwidth is exactly what we set using Dummynet [19].

2) Video clip and time-varying network bandwidth: We download and use a 10 minute video clip from Youtube website [1] that has 5 different versions, the 1st, 4th and 5th are MP4 type while the 2nd and the 3rd are FLV type. We name the versions as 1 to 5 with the quality increasing and average bit rate increasing. The initial version is version 3, with neither too high nor too low quality.

We divide the 10 minute playing time into 60 intervals, i.e., each interval lasts 10 seconds. We use a network bandwidth trace as shown in Figure 6(a) to evaluate the application performance under the fuzzy controller. During the intervals 1-20, the available network bandwidth is 5Mbps; during the intervals 21-40, the available network bandwidth drops to 1Mbps; during the intervals 41-60, the available network bandwidth goes back to 5Mbps. As we can see from Figure 6(a), the actual bandwidth utilization is always below the available bandwidth.

For comparison purpose, we also implement a traditional controller which is used in one of the state-of-the-art methods [20]. The controller will upgrade the version immediately when the network bandwidth utilization is below the setting point and downgrade the version when the network bandwidth utilization is above the setting point. The bandwidth availability and usage, bandwidth setting and utilization and adaptive video versions under NBS and the traditional controller are shown in Figure 6 and Figure 7, respectively.

B. Adaptive version switcher

Figure 6(a) shows that the network bandwidth usage is always within the limit while the available network bandwidth is fluctuating.

More specifically, Figure 6(b) shows the results for the bandwidth utilization and our bandwidth utilization setting point, which is 80%. During the intervals from 1 to 20 when the available bandwidth is 5Mbps, the fuzzy controller switches from the high bit rate version to the high bit rate version and keeps the bandwidth utilization around the setting point. During the intervals from 21 to 40 when the available bandwidth decreases to 1Mbps, the fuzzy controller switches from the high bit rate version 5 to the low bit rate versions 1 and 2 and keeps the bandwidth utilization around the setting point.

Figure 6(c) shows the results for the video version that is chosen along the time. During the intervals from 1 to 20 when the available bandwidth is 5Mbps, the fuzzy controller starts from version 3 and discovers the low bandwidth utilization around 50%. It then decides to switch to a version that has better quality and higher bit rate. It switches to the version 4 and then switches to version 5. During the intervals from 21 to 40 when the available bandwidth drops to 1Mbps, the fuzzy controller discovers the high bandwidth utilization around 100% if it stays with version 5. It then decides to switch to a version that has a lower bit rate. As we can see that, it switches to the version 4 and then switches to version 5. During the intervals from 41 to 60 when the available bandwidth increases to 5Mbps again, the fuzzy controller discovers the low bandwidth utilization around 20% if it stays with lower versions like 1 and 2. It then decides to switch to a version that has better quality and higher bit rate. As we can see that, it switches to the version 3 and then switches to version 4, 5.

We observe similar result when we use a traditional controller as shown in Figure 7. During the intervals from 1 to 20 when the available bandwidth is 5Mbps, the controller
starts from version 2 and discovers the low bandwidth utilization around 50%. It then upgrades the version until version 5. However, due to the fluctuation of the network bandwidth consumption, it switches frequently between versions 4 and 5. During the intervals from 21 to 40 when the available bandwidth drops to 1Mbps, the controller downgrades the version until 2.

C. Responsiveness and stability

We look into more details in Figure 6(c) and Figure 7(c). It takes only 3 intervals (intervals 21 to 23) for NBS to switch from the version with the highest bit rate to the lowest version when the available bandwidth drops from 5Mbps to 1Mbps. This shows a good responsiveness for the controller when the available bandwidth drops. However, compared with the case when the bandwidth drops, it takes 5 intervals (intervals 1 to 5) and 10 intervals (intervals 41 to 51) for the controller to switch from the version with the lowest bit rate to the highest one when the available bandwidth increases from 1Mbps to 5Mbps. This shows a fair responsiveness for the controller when the available bandwidth increases. This difference in response time reflects that the controller acts more conservatively when available bandwidth increases compared with the case when available bandwidth drops. We can also find that, when the available bandwidth is 5Mbps, version 5 is a good choice for the user because its bit rate will provide a network bandwidth utilization around 80%. However, when the available bandwidth is only 1Mbps, there is no fixed candidate version for the user to provide a network bandwidth utilization around 80%. Version 2 will provide a network bandwidth utilization near 100% and version 1 will provide network bandwidth utilization below 80%. The controller will switch between these two versions. This shows that NBS is more stable when the available bandwidth is enough, e.g., 5Mbps and more unstable when the available bandwidth is not enough.

Compared with NBS, the traditional controller has a slightly better responsiveness but unacceptable instability. It takes 3 intervals (intervals 21 to 23) for the controller to switch from the version with the highest bit rate to the lowest version when the available bandwidth drops and it takes 3 intervals (intervals 1 to 3) and 5 intervals (intervals 41 to 45) for the controller to switch from the version with the lowest bit rate to the highest one when the available bandwidth increases. However, the traditional controller is too sensitive to the changes in the network bandwidth utilization and instable in the control actions. For example, it changes between version 1 and version 2 during intervals 23 to 42 and between version 4 and version 5 during intervals 50 to 60 back and forth. The sensitive actions will seriously harm the perceptual quality.

V. RELATED WORK

There are a plethora of previous works on adaptive multimedia applications [21], [22], from different points of view and using different methods. Akhshabi et al. [23] experimentally evaluate two major commercial players and one open source player about the rate-adaptation mechanisms of adaptive streaming. Prangl et al. [24] propose an approach to increase the perceived QoS for TCP-based video streaming over best effort networks. Their approach is based on the continuous adaptation (transcoding) of the video stream and taking into account TCP throughput measurements at the server side. The research reported in [25] adapts the
streaming processes by transmitting only some components of the multimedia streams. Aggarwal et al. [26] use an Earliest Deadline First (EDF)-like scheduler to schedule the transmission of video chunks according to their deadlines using multicast. Alcock et al. [27] investigate into the application flow control technique that is utilized by YouTube servers. Most of these works use various methods, e.g., selective transmitting [25], (EDF)-like scheduler [26] and flow control technique [27] on the server side.

Muntean et al. [20] propose an adaptive method based on classic feedback control. In their work, during transmission, the server can switch between different quality versions of the same multimedia content at certain checkpoints to modify the quality of the overall streaming process, and therefore the transferred quantity of data. However, in our work, we implement the switcher on the user side as the user shares the network bandwidth with others. Moreover, we have a fuzzy model and a rigorous fuzzy controller based on the model in our work rather than a traditional feedback control approach. The experiment result also shows that our method outperforms the traditional method by obtaining stable versions.

Razavi et al. [28] present fuzzy logic control of Automatic Repeat Request (ARQ), based on send buffer fullness and the head of line packet’s deadline. Jammeh et al. [29] propose a fuzzy-logic congestion controller which changes the sending rate of a video transcoder using packet dispersion rather than feedback of packet loss. Vehkaperä et al. [30] propose a fuzzy logic-based application layer controlling algorithm for scalable video streaming to adjust the bit rate of transmitted video utilizing cross-layer feedback information collected from the physical, network, and application layers. Our paper and the above work show that fuzzy logic is appropriate and promising to be taken as a theory basis in the video streaming area.

VI. CONCLUSIONS

Achieving the best quality of service [31] for video streaming application with time-varying network bandwidth is an important problem for both the video streaming application provider and the users. In this paper, we build NBS, i.e., a network bandwidth-aware version switcher for video streaming application. It uses a fuzzy controller to help the users to decide the best version that fits the users’ network bandwidth capability without the users’ interference. Our experimental results show the effectiveness of our approach. Future work will focus on (1) examining the robustness of our controller using a mobile device, e.g., using a wireless phone instead of a physical machine; (2) considering more factors (e.g., available power in the mobile device) which may also affect the optimal choice of streaming video version. An interesting research direction would be how to adapt to mobile clients’ energy levels for dynamic switching of video streams. For instance, when a mobile client is low in battery, the optimal choice of streaming video version may be a low quality one to reduce the energy consumption at the client side.

ACKNOWLEDGMENT

This research has been partially funded by National Science Foundation by IUCRC/FRP (1127904), CISE/CNS (1138666), RAPID (1138666), CISE/CRI (0855180), NetSE (0905493) programs, and gifts, grants, or contracts from DARPA/I2O, Singapore Government, Fujitsu Labs, Wipro Applied Research, and Georgia Tech Foundation through the John P. Imlay, Jr. Chair endowment. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

REFERENCES


[18] “Java Bindings for VideoLAN,”
http://code.google.com/p/vlcj/.

[19] “Dummynet,”
http://info.iet.unipi.it/~luligi/dummynet/.


